AIAA Modeling and Simulation Technologies Conference and Exhibit

A
COLLECTION
OF
TECHNICAL
PAPERS

Boston, Massachusetts
August 10–12, 1998

For permission to copy or republish, contact:
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive
Suite 500
Reston, VA 20191-4344

ISBN
1-56347-271-6

Printed in the U.S.A.

AIAA Modeling and Simulation Technologies Conference and Exhibit
Boston, MA / August 10–12, 1998

TABLE OF CONTENTS

Paper No. Title and Author Page Number

Session 11-MST-1 Aircraft Dynamics, Modeling and Performance

Session 12-MST-2 Networked Technologies and Applications

Session 13-MST-3 Human Factors, Cueing Systems, and Environmental Modeling

WD—Withdrawn NA—Not Available

Session 14-MST-4 Simulator/Flight Training Device Development & Program Overviews

Session 31-MST-5 Workshop: DoD's High Level Architecture (HLA) for M&S

Session 46-MST-7 Vehicle Dynamic Modeling

Session 47-MST-8 Motion Systems

WD—Withdrawn NA—Not Available

Session 48-MST-9 Low Cost Simulation

Session 49-MST-10 Space Systems Simulation

Session 81-MST-13 Simulation Standards and Reuse

Session 82-MST-14 Training Testing, Verification, and Validation

WD—Withdrawn NA—Not Available

Session 83-MST-15 Avionics, Weapons and Engagement Modeling and Simulation

AIAA-98-4159

GENESIS NON-REALTIME SIMULATION OF A TAILLESS AIRCRAFT

Phillip D. McKeehen*
Air Force Research Laboratory(AFRL/VACC)
Wright-Patterson Air Force Base, Ohio 45433-7521

ABSTRACT

This paper discusses the use of GENeral Environment for the Simulation of Integrated Systems(GENESIS) in the non-realtime 6 DOF simulation of a tailless 65-degree delta-like wing aircraft with a large and innovative suite of control effectors. The discussion topics include: an overview of the Fortran 77-based GENESIS software's structure and basic functionality; the two aerodynamic models that have been used for the vehicle's simulation; the second aerodynamic model's use of the Kalviste method of implementing rotary balance wind tunnel data for improved high angle-of-attack flight simulation; a brief description of a modem control law design technique used with the first aerodynamic model of the vehicle; examples of simulation output time-histories for simple low and high angle-of-attack maneuvers.

INTRODUCTION

A major area of current research in the aerospace community is concerned with the analysis and design of vehicles without vertical tails. While these vehicles offer certain advantages over conventional aircraft with tails (e.g. reduced observability), they also present considerable challenges to the designer. Lack of a vertical tail forces the flight control system designer to use other yaw control effectors for enabling the aircraft to still possess the requisite agility for combat situations.

Reference [1] describes the development of robust flight control laws and their evaluation using simulation of a tailless fighter design currently being studied by the Air Force Research Laboratory's Control Dynamics Branch and Lockheed-Martin Tactical Aircraft Systems (LMTAS). The tailless aircraft design evolved from a research program intended to explore new control concepts for effectors such as all-moving wing tips, differential leading-edge flaps, spoiler-slot deflectors and pitch and yaw thrust vectoring(see Fig 1). The control laws described in [1] were designed for a configuration which included all-moving wing tips, pitch and yaw thrust vectoring, pitch flaps and elevons on a 65-degree delta-like wing configuration. The resulting aircraft simulation was implemented in GENESIS. The design evaluation noted in [1] showed acceptable flying qualities, stability and robustness to inherent aircraft modeling uncertainties within the design envelope of machs ranging from .3 to .5 and altitudes ranging from 10,000ft to 20,000ft.

Following the flight control law design documented in [1], different control laws have been designed for a similarly configured version of this vehicle, which also included differential outboard leading-edge flaps and spoiler slot-deflectors. Additionally, a new aerodynamic model, which includes rotary balance wind tunnel data has recently been developed and integrated with the rest of the simulation software.

The primary subject of this paper is the description of the two aerodynamic models and their implementation in the GENESIS software. For the remainder of the paper these two distinct models and the executable simulation software modules associated with them will be referred to as Innovative Control Effectors (ICE) I and II.

THE GENESIS SOFTWARE ENVIRONMENT

GENESIS is a proprietary Fortran 77 software product of the Northrop-Grumman Corporation[2]. It is supported on a number of different platforms: VAX/VMS, IBM/TSO, SUN/UNIX, SGI/UNIX, HP/UNIX, and PC/DOS. GENESIS is capable of performing in either a non-realtime or real-time capacity. Its structure and functionality are described in detail in [2].

* Senior Member AIAA

The GENESIS software can be used to simulate any time-varying system and has been used to simulate guided weapons, mass transit vehicles and automobiles. Aerospace simulation applications at AFRL include a supermaneuverable F/A-18, a hypervelocity vehicle, the F-15 / SMTD and various configurations of the VISTA/F-16. Although generic, the software evolved from aircraft simulation applications. For this reason, many standard aircraft simulation features are built-in. For example, modules exist for the 6 DOF rigid body aircraft equations of motion, a standard atmosphere model and standard disturbance models. Only such vehicle specific subroutines as actuators, propulsion system, airframe, landing gear and control laws or various databases (aerodynamics and propulsion) must be developed by the user (see Fig 2).

Besides the standard components, GENESIS has simulation core software comprised of built-in routines for linear model extraction, extensive trimming capabilities, database management and many utility functions (control system filters, matrix manipulations, etc.). The extensive and powerful user interface incorporates a wide range of commands that allow the performance of a diverse array of system analyses and a variety of simulation development debugging tasks.

The flexibility and ease of use of the software structure shown in Fig 2 is further enhanced by the hierarchical use of distinct libraries for configuration management (see Fig 3). At the lowest priority level is the simulation library, containing generic simulation subroutines which can be used by any GENESIS simulation (tailless aircraft, VISTA/F-16, F-15/SMTD etc). The next higher level library is the project library. Here one library would be specifically for the tailless aircraft, a separate one for the F-16 and a separate one for the F-15/SMTD. The highest priority library is the user project library. Here a user may have additional routines or slightly modified routines for a specific vehicle simulation which take precedence over the lower level routines. In this way a user may still run the current version of a particular simulation while simultaneously developing and debugging a new version of it and maintaining complete configuration control.

GENESIS ICE I AERODYNAMIC MODEL

The extended suite of control effectors gives the capability to have multiple configurations of this tailless vehicle. Thus, in addition to the aerodynamic controls (pitch-flaps, elevons, and All-Moving- Tips (AMT)) used in the configuration simulated in [1], the database also contains stability and control data for differential inboard and outboard Leading-Edge-Flaps (LEF), spoilers and slot deflectors (SSD). Aerodynamic forces and moments are calculated based on the use of traditional linear methods of estimating stability and control derivatives. These derivatives and basic force and moment data are then used as coefficients in equations which build up the total aerodynamic force or moment (pitching moment, rolling moment, yawing moment, side force, normal force, and axial force). GENESIS subroutine X40AERO.f contains the code for extraction of all basic data and individual control surface incremental data, together with all equations to build up the total force and moment coefficients. As an example, the calculation of pitching moment is done from equations of the general form:

$$C_m = C_{m_b} + \delta C_m + (C_{m_q} q \bar{c}) / (2v) + C_{m_{c\text{-}ssd}}$$

$$\delta C_m = \sum_{i=1}^{n} e_i \, \delta C_{m_i}$$

where:

C_m = total pitching moment coefficient

C_{m_b} = basic pitching moment coefficient of airframe, controls undeflected

δC_m = total pitching moment contribution from control surfaces

n = number of control effectors contributing to δC_m

e_i = individual surface control effectiveness ($e_i = 1$ indicates same effectiveness as in wind tunnel tests)

δC_{m_i} = pitching moment coefficient of individual surface

C_{m_q} = pitch damping derivative

q = pitch rate

$\bar{c}$ =mean aerodynamic chord

v = free stream velocity

$C_{m_{e\text{-}ssd}}$ = contribution to pitching moment due to interaction of elevon and SSD

The other five aerodynamic force or moment coefficients are calculated similarly. Depending on the particular configuration simulated, certain control effectors may not be used. Figures 4-10 show examples of the basic pitching moment and the contribution to pitching moment of the pitch flaps, elevons, SSD, AMT and differential outboard LEF at lowspeed (Mach=.3). Note that the plots are given for left surface deflection. A similar contribution would be given for the corresponding right surfaces. Note that a single table of values is given for symmetric pitch flap. The interaction of slot deflector and elevon is also modeled at low speeds in the ICE I aerodynamic model. Dynamic derivatives were predicted using High Angle of Attack Stability and Control (HASC) and DYNAMIC aerodynamic prediction codes. The aerodynamic model is valid in the range from Mach=.3 to Mach=2.16, angle-of-attack from -2.5 to 45 degrees and sideslip from -10 to 10 degrees. The model also includes aeroelastic effects

GENESIS has automatic linear interpolation between breakpoints of all database curves. In addition, since the ICE I aerodynamic base is segmented into low speed (Mach=.3) and high speed(mach=.6 or higher) data, X40AERO.f contains code to linearly interpolate the total aerodynamic coefficients for mach numbers between .5 and .6. Following some discussion of the engine model and flight control law design, examples of simple maneuvers using these pitch axis aerodynamics and the rest of the aerodynamic model are shown.

ICE I ENGINE MODEL

The simulation uses a model of the F-110-GE-229 engine. The engine thrust is a function of power level angle (PLA), Mach number and altitude. Figure 11 shows samples of the thrust generated by this engine at PLA = 87.5deg, where idle power is given by PLA = 20 deg and max afterburner by PLA=127deg.

ICE I CONTROL LAWS

Subsequent to the control law design discussed in [1], another control law design for the ICE I configuration was done by Buffington[10,11]. This design uses the method of feedback linearization[6] or dynamic inversion[7,8] to stabilize the short period and dutch-roll modes and provide decoupled first order response to body-axis rate commands. Readers interested in this approach to control law design are encouraged to see also [5,9] for more detailed information.

ICE I SIMULATION EXAMPLES

This section presents two straightforward examples of the use of the ICE I configuration and control laws described above in simulated pitch and roll maneuvers. In the first simple maneuver, the ICE vehicle was trimmed straight and level at Mach=.4 and altitude 15,000ft. A 15 degree/sec pitch rate command was then issued and held for 3 seconds, at which time a 0 deg/sec pitch rate command was issued. Figure 12 shows the pitch rate command and response. Figure 13 shows the angle-of-attack time history corresponding to the pitch rate command and pitch rate time history.

In the second maneuver, the same pitch rate command is issued, but at t=4 seconds (with angle-of-attack=30 degrees), a 10 deg/sec roll rate command is issued for 2 seconds, followed by a -10 degree/sec roll rate command for 2.5 seconds. Figure 14 shows the time histories for pitch rate and roll rate associated with this maneuver. Figure 15 shows the corresponding angle-of-attack and bank angle time histories. Note that these two simple maneuvers were chosen purely for academic interest. Certainly, they are not meant to be indicative of realistic combat maneuvers, which of course, can be simulated with this software.

GENESIS ICE II AERODYNAMIC MODEL

A completely revised and higher fidelity aerodynamic model (ICE II) was developed in GENESIS by James Simon and the author in late 1997, based on three additional wind tunnel tests of the vehicle. The aerodynamic control effectors included with the ICE II model are: elevons, pitch flaps, differential inboard and outboard LEF, AMTs, and SSDs. Increased model fidelity is achieved over the ICE I aerodynamic model in two ways. First, ICE I wind tunnel testing included no forced oscillation tests for dynamic derivatives nor rotary balance tests, whereas ICE II data included results from both of these types of wind tunnel tests. Second, ICE I aerodynamics include only the interaction effects of the spoiler on the elevon. ICE II includes the interaction effects of AMT with outboard LEF, elevon with AMT, and Mach effects on SSD and AMT. The wind tunnel tests conducted for ICE II covered an angle-of-attack range of -3 to 90 degrees, sideslip range of -30 to 30 degrees, and Mach number range of .3 to 2.16. This Mach number range is the same as for ICE I, but for ICE II, a single segment of data includes all Mach numbers, while ICE I has two segments(separate low and high Mach number segments).

Another fundamental difference between ICE I and ICE II aerodynamic modeling and its GENESIS implementation is that the GENESIS database management system has a limit of four independent variables. This was surpassed for the ICE II increments due to symmetric pitch-flaps and also outboard LEF deflection. For the LEF increments, this implementation problem was easily solved by construction of two separate segments of this increment, based on the ranges of the independent variables for which the data was collected. Segment 1 was constructed for Mach =.3 only and then became a function of angle-of-attack, sideslip, outboard LEF deflection, and inboard LEF deflection. Segment 2 was constructed at Mach = .6, .9 and 1.2, but was a function of angle-of-attack, sideslip and outboard LEF deflection only (not inboard LEF deflection). Segments 1 and 2 dependent variable (i.e. Cm , Cn, etc.) values attributed to outboard LEF deflection were then found by using typical linear interpolation methods between breakpoints in segment 2 and between segment 1 and 2 increments. For the increments due to pitch-flap deflection, three separate segments of the increment were constructed. A low speed segment was constructed for Mach=.3 only, as a function of angle-of-attack (-2.5 to 90 degrees), left and right SSD deflection and pitch-flap deflection. A medium speed segment was constructed as a function of Mach (.6,.9 and 1.2), angle-of-attack (0 to 30 degrees) and pitch-flap deflection (-30, 0, 30 degrees). The high speed segment had the same angle-of-attack breakpoints, with Mach numbers (1.6, 2.0, 2.16), and pitch-flap settings of -30, -10, 0, 10, 30 degrees. Again linear interpolation was used between segmented aerodynamic coefficient values; but, for ICE II it is values of the increments (Cm, Cn, etc.) due to pitch-flaps that are so interpolated between segments. (This is in contrast to ICE I where the *total coefficient* including all increments is interpolated between the low Mach number and high Mach number value). This ICE II interpolation scheme was used in lieu of an initial interpolation over several breakpoints of data to produce a fully "rectangular" table of breakpoints for the increments to the coefficients due to pitch-flap deflection.

High angle-of-attack flight typically involves rolling about the velocity vector instead of body axis rolls. To be able to better simulate this type of roll and also to predict spins, ICE II wind tunnel testing included rotary balance tests, in which the wind tunnel model is rotated about the velocity vector. It also included forced oscillation tests (oscillation about a body axis) to yield improved aerodynamic data for the dynamic derivatives (roll due to roll rate, roll due to yaw rate, yaw due to roll rate, etc.). The GENESIS implementation of this rotary and forced oscillation wind tunnel test data uses the Kalviste[3] method of mathematically processing this data. Although the current state of the art in the use of mathematical algorithms to process rotary balance data does not include an industry-wide accepted standard procedure, the method explained here and used in the ICE II vehicle simulation has been used by several major aerospace industry organizations. The method may be stated as follows:

For a general 6 DOF simulation, the total rotation vector Ω is defined in terms of the three body axis rates (P,Q,R). If ω is defined as the component of this total rotation vector along the velocity vector, then the following algorithm is used to calculate the dynamic terms in the aerodynamic build-up equations. Consider the equations

$$P_{mod} = P - \omega \cos\alpha \cos\beta \qquad (1)$$

$$Q_{mod} = Q - \omega \sin\beta \qquad (2)$$

$$R_{mod} = R - \omega \sin\alpha \cos\beta \qquad (3)$$

First compute ω from equation (1) with Pmod set equal to zero. This value of ω is used with equations (2) and (3) to compute Qmod and Rmod. If the computed values of Qmod and Rmod are less than, and of the same sign as Q and R respectively, then the three components of rotation rate are taken to be Qmod, Rmod, and ω. If either of these conditions is not satisfied, the same procedure is then applied to equation (2). If both conditions are not met for the application to equation (2) the procedure is applied to equation (3). If the test still fails (Pmod not less than P or Qmod not less than Q or Pmod and Qmod not both of the same sign as P and Q), then the result is that the modified components of rate are just set equal to the body axis rates P,Q,and R and ω is set to 0 for the calculation of the dynamic part of the aerodynamic coefficient build-up equations. As an example, once the modified rates are computed, the dynamic term in the equation for rolling moment coefficient becomes

$$C_{l\,dyn} = C_{l_p} P_{mod}(b/2v) + C_{l_r} R_{mod}(b/2v) + \delta$$

where b=wing span, v=velocity, and δ is an incremental term dependent on α, β and steady rotation rate $\omega b/2v$. The remainder of the aerodynamic build-up equations (static component and increment due to control surface deflection) are similar to those of ICE I. These modified rates are used *only* in the calculation of the dynamic terms of the aerodynamic build-up equations. The actual aircraft rotational rates P,Q,R are used in the rest of the aircraft dynamic simulation (equations of motion, etc.).

At the current time (May 1998), new flight control laws have not yet been designed in GENESIS for the ICE vehicle described here using the ICE II aerodynamic model. It is anticipated that such control laws will indeed be designed in the near future.

CONCLUSIONS

This paper has described a simulation of a tailless aircraft which resides in the GENESIS simulation software environment. Two distinct aerodynamic models have been developed for the vehicle, which has an expanded suite of aerodynamic control effectors and also uses pitch and yaw thrust vectoring. Several control law designs have been done based on the original aerodynamic model of the vehicle. It is anticipated that new control laws will soon be designed for the second version of the aerodynamic model, which includes rotary balance and forced oscillation wind tunnel test data and the modeling of additional aerodynamic interaction effects. The GENESIS tailless aircraft simulation is also currently being implemented in the Air Force Research Laboratory's piloted simulation facility. Later this year, the author and his co-workers plan to perform a piloted simulation study of this vehicle to investigate control allocation / optimization methodologies and their impact on aircraft handling qualities.

ACKNOWLEDGMENTS

The author wishes to acknowledge the contributions made by several Control Dynamics Branch engineers in regard to the tailless aircraft simulation and its associated software described in this paper. Mr Ahntuan Ngo developed the GENESIS ICE I aerodynamic model. Dr James Buffington designed the GENESIS ICE I control laws. Mr James Simon developed part of the GENESIS ICE II aerodynamic database. Mr William Gillard was the project engineer responsible for monitoring the collection of both ICE I and ICE II aerodynamic data in the wind tunnel. Their excellent work was essential to the development of this tailless aircraft simulation.

REFERENCES

[1] Ngo, A.D., Riegelsperger,W.C., Banda, S.S., "Robust Flight Control Law Design for a Tailless Airplane", Proc. AIAA Guidance,Navigation, and Control Conference, San Diego, CA, July 1996.

[2] Wong,J.P. et al "Companion Guide to GENESIS Software Version 2.2", Vol I-User's Guide, Vol II - Programmer's Guide, Vol III- Linearization Guide, Northrop Corporation B-2 Division, September 1993.

[3] Kalviste, J., "Use of Rotary Balance and Forced Oscillation Test Data in a Six Degrees of Freedom Simulation", Proc. AIAA Atmospheric Flight Mechanics Conference, San Diego, CA, August 1982.

[4] Gillard, W.J., Dorsett, K.M., "Directional Control for Tailless Aircraft Using All Moving Wing Tips", Proc. AIAA Atmospheric Flight Mechanics Conference, New Orleans, LA, August 1997.

[5] Kocurek, J.N., Durham, W.C., "Dynamic Inversion and Model-Following Flight Control: A Comparison of Performance Robustness", Proc. AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, July 1997.

[6] Isodori, A., "*Nonlinear Control Systems*", Springer-Verlag, 1989.

[7] Enns, D., Bugajski, D., Hendrick, R., Stein, G., "Dynamic Inversion: an Evolving Methodology For Flight Control Design", *International Journal of Control* Vol. 59, No. 1, 1994, pp. 71-91.

[8] Honeywell Technology Center, "Application of Multivariable Control Theory to Aircraft Control

Laws. Final Report: Multivariable Control Design Guidelines", WL-TR-96-3099, May 1996.

[9] Morton, B., Enns, D., Zhang, B.Y., "Stability of Dynamic Inversion Control Laws Applied to Nonlinear Aircraft Pitch Axis Models", IMA Preprint Series # 1245, July 1994.

[10] Buffington, J.M., "Control Design and Analysis for Systems with Redundant Limited Controls", Ph. D. Thesis, University of Minnesota, 1996.

[11] McKeehen, P.D., Myatt, J.H., Grismer, D.S., Buffington, J.M., "Simulation of a Tailless Aircraft Using Nonlinear Indicial Response Aerodynamic Modeling Methods", Proc. AIAA Modeling and Simulation Technologies Conference, New Orleans, LA, Aug 1997.

FIGURES

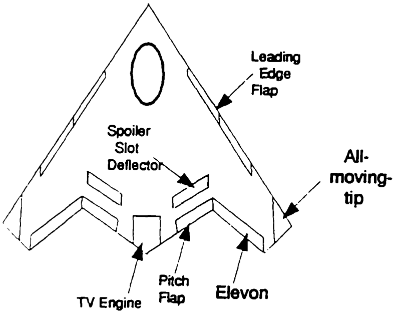


Fig 1 - Tailless Aircraft Configuration

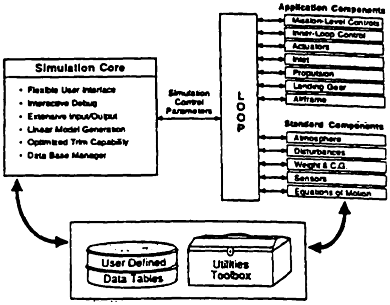


Fig 2 - GENESIS Structure

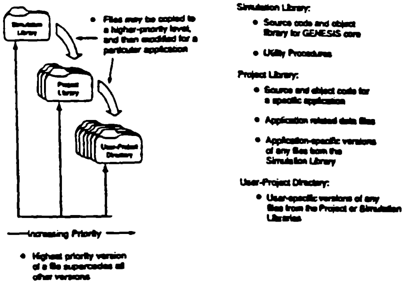

Simulation Library
Project Library
User-Project Directory
• Files may be copied to a higher-priority level, and then modified for a particular application
Increasing Priority
• Highest priority version of a file supercedes all other versions
Simulation Library:
• Source code and object library for GENESIS core
• Utility Procedures
Project Library:
• Source and object code for a specific application
• Application related data files
• Application-specific versions of any files from the Simulation Library
User-Project Directory:
• User-specific versions of any files from the Project or Simulation Libraries

Fig 3 - GENESIS Configuration Management

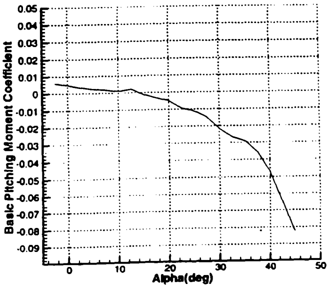


Fig 4 - ICE I Basic C_m
(Beta = 0 deg)

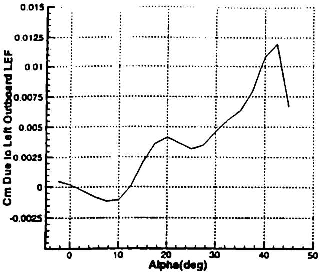


Fig 5 - ICE I C_m Due to L. Outboard LEF
(Beta = 0 deg / OBLEF=40deg)

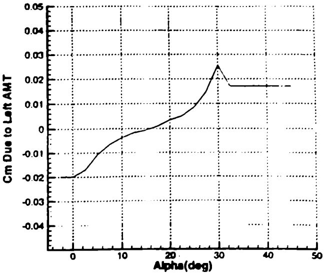


Fig 8 - ICE I C_m Due to Left AMT
(L. AMT=30deg / Beta=10deg)

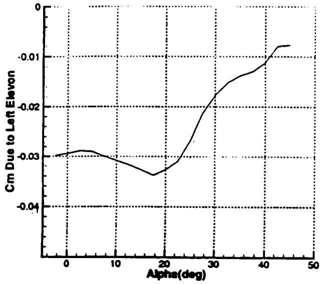


Fig 6 - ICE I C_m Due to L. Elevon
(L. Ele=30deg / L. Spoil = 0deg / OBLEF=0deg)

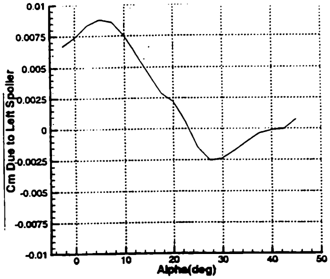


Fig 9 - ICE I C_m Due to Left Spoiler
(L. Spoil=30deg / L. OBLEF=40deg)

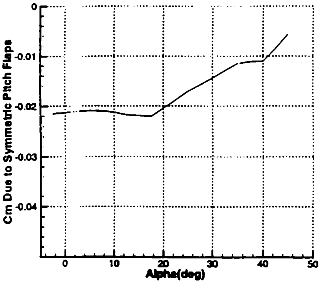


Fig 7 - ICE I C_m Due to Symm Pitch Flaps
(P.Flaps=30deg / L. Spoil=0deg)

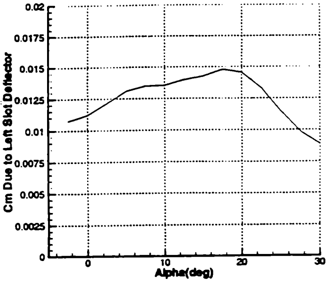


Fig 10 - ICE I C_m Due to Left Deflector
(L. defl =30deg / L. Spoil=60deg)

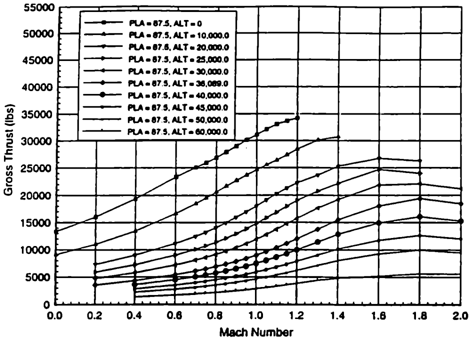


Fig 11 - Engine Model Thrust Curve Examples

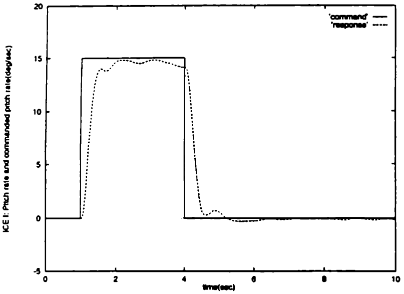


Fig 12 - 15 Deg/sec Pitch Rate Command at Mach=.4 and Alt=15,000ft

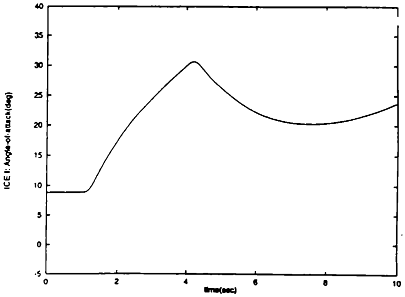


Fig 13 - Angle-of-attack for Pitch Rate Command at Mach=.4 and Alt=15,000ft

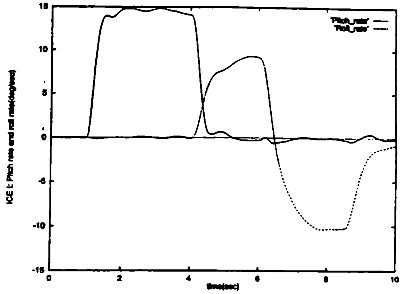


Fig 14 - Pitch/Roll Rate Commands at Mach=.4 and Alt=15,000ft

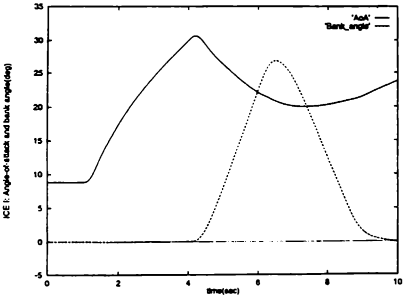


Fig 15 - AOA and Bank angle for Pitch/Roll commands at Mach=.4 and Alt=15,000ft

AIAA-98-4160

SIMULATION OF A F/A-18 E/F DROP MODEL USING THE LaSRS++ FRAMEWORK

Kevin Cunningham*, P. Sean Kenney, Richard A. Leslie
David W. Geyer, Michael M. Madden*, Patricia C. Glaab

Unisys Corporation
NASA Langley Research Center
MS 169
Hampton, VA 23681

Abstract

A simulation of a 22% dynamically scaled F/A-18 E/F Drop Model was successfully developed within the Langley Standard Real-Time Simulation in C++ (LaSRS++) framework. Development in the LaSRS++ framework is done using object-oriented analysis, design and programming techniques. Common software design patterns are also used. Development using the LaSRS++ framework promotes the development of a simulation which is inherently maintainable, extensible, reliable and computationally efficient.

Introduction

The goals of maintainability, extensibility and reliability are certainly common to all software development efforts. The use of object-oriented analysis, design and programming techniques to meet these goals is well established.[1–4] What is not well established is the application of these techniques to the development of a piloted flight simulation that must necessarily operate in a hard real-time environment. These techniques were used to develop a maintainable, extensible and reliable F/A-18 E/F Drop Model Simulation for the Simulation Systems Branch at NASA Langley Research Center. Furthermore, the unusually strict demands for computational efficiency imposed by the simulation of dynamically scaled models were met with ease.

The F/A-18 E/F Drop Model Simulation was developed in support of the Drop Model Program at NASA Langley Research Center. The 22% dynamically scaled model is flight tested to support aerodynamic and flight control research interests. As part of the test technique,[5] the unpowered model is carried to its release altitude by helicopter. After release from the helicopter, a pilot on the ground uses down-linked drop model cockpit video and sensor data to fly the flight profile. Commands from the crew station, along with the down-linked data are used by the ground based flight control computer to calculate control surface actuator commands. The resulting commands are up-linked to the model. At the end of the flight, a parachute is deployed and the model recovered.

For the drop model to achieve an accurate representation of the full scale aircraft's flight dynamic characteristics, dynamic scaling techniques must be applied to the model.[6] The geometric and mass properties of the model are dynamically scaled by matching the Froude number. The same scaling technique must be applied to the temporally dependent aspects of the flight control laws as well as the sensed data used in the control laws. The scaling technique dictates that the model's time scale is inversely proportional to the square root of

* Senior Member, AIAA

the geometric scale factor. Thus, for a 22% dynamically scaled model, the control laws must run at a frame rate over 200% faster than that of the full-scale aircraft. For this reason, the implementation of dynamically scaled control laws imposes unusually high demands for computational efficiency.

This F/A-18 E/F Drop Model Simulation is primarily used for evaluation and tuning of control laws, flight profile planning and crew training. Additionally, the implementation of the flight control laws used by the simulation serve as a verification model for the Drop Model Program's independent implementation of the same laws. The simulation, developed using the GNU C++ compiler, supports pilot-in-the-loop, synchronous real-time operations with integration rates in excess of 300 Hz on SGI Onyx systems at NASA Langley Research Center.

Fundamental Concepts

The fundamental concepts behind the LaSRS++ framework[7] are:

- simplification of behavior (via abstraction)
- protection of data (via encapsulation).

The use of object-oriented C++ and a few simple implementation guidelines within the framework create an environment where these concepts are not only simply allowed, but fostered. Simplification of behavior is primarily achieved by abstraction of detail. When details are handled elsewhere, the user need not be distracted by the complexities at hand. In object-oriented design, this takes on a layered form. Once the desired levels of abstraction are decided upon, these various levels of complexity are represented by separate classes.

Classes are the fundamental building blocks in C++. A class usually represents a single concept or physical entity. A developer specifies the behaviors (functions) and the data that a class will contain. An object is a particular instance of a class. An object named "Titanic" would likely be a particular instance of a doomed ocean-liner class. In this regard one can think of a class as they would think of an intrinsic data type (int, float, double). One object that creates a second object is said to contain that object. This is often referred to as a "has a" relationship. (e.g. An F/A-18E has a flight control system.)

Inheritance is the mechanism by which the details from one layer are added to another. The inheritances may be chained together. Inheritance is often referred to as an "is a" relationship (an F/A-18 is an airplane). Thus, a class representing an F/A-18 would not contain all the aspects that are common to all aircraft, just the aspects that make an F/A-18 unique. Inheritance is the primary means for the abstraction of complexity.

Polymorphism is one of the most powerful features of object-oriented languages. It is the means by which a behavior declared in a parent class may be defined differently by each child class. It allows different behavior to be realized via a common interface. This simplifies software and promotes the use of common interfaces.

The C++ language provides a means to allow or disallow a client's access to the behaviors and data in a class. Public access (any client has access) and private access (no client has access) provide the two extremes of data/function protection. A significant feature of the LaSRS++ framework is that all data, in all classes of the framework is private. Containment ("has a") or inheritance ("is a") relationships are not permitted to break this data encapsulation. If the developer of a class wishes to allow outside access to the data in a class, accessor functions with the appropriate level of protection (public or protected) must be written to reference the data. The data access provided may be limited to read-only or, if required, both read and write. In this manner, the LaSRS++ framework provides excellent data protection from accidental corruption through the use of strong encapsulation.

The Framework

The F/A-18 E/F Drop Model Simulation was developed within the LaSRS++ framework. A framework is a collection of classes which are used to build various products. This framework is kept under strict configuration control. The classes contained therein are carefully reviewed with regard to both design and implementation.

Classes in the LaSRS++ framework[7] can be divided into two major categories. The first is very general in nature and serves as a "toolbox" which contains classes for:

- filtering
- function table lookups
- vector and matrix operations
- statistical and random variable calculations
- memory management[8]
- input - output capabilities[9]
- data conversions

The second is oriented more toward simulation and contains classes relating to:

- equations of motion
- propulsion systems
- flight control systems
- weapons systems
- aerodynamic modeling
- environmental modeling
- mass property modeling
- multi-vehicle, multi-cpu simulation[10]
- hardware interfacing[11]
- navigation systems
- trim capabilities

Using The Framework

Two classes of interest in the framework are named Vehicle and Aircraft. These classes are shown, along with their relationship to an F/A-18 E/F Drop Model class in Figure 1.

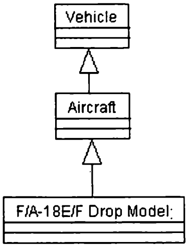


Figure 1: Inheritance in action

The diagram presented in Figure 1 is a class diagram using the Unified Modeling Language.[12] It makes the simple statement that the F/A-18 E/F Drop Model class inherits from Aircraft which inherits from Vehicle. (An F/A-18 E/F Drop Model "is a" Aircraft, which "is a" Vehicle.) The Vehicle class contains data such as acceleration, velocity and position vectors. It also contains functions for the calculation of these quantities. Aircraft contains data such as angle of attack, side slip, calibrated airspeed and functions to calculate these quantities.

The F/A-18 E/F Drop Model class need only take responsibility for registering its unique attributes with the Aircraft and Vehicle classes. Since the data in those classes only allows private access, this is implemented using in-lined mutator functions. These mutator functions would typically have only protected access (only a sub-class may use the function). The use of the C++ `inline` keyword plus compiler optimizations make the use of protected mutator functions a very efficient means of controlling access to data. In addition to defining its attributes, the F/A-18 E/F Drop Model class must also define its unique behaviors. In practice, this primarily consists of a few behaviors that update the forces and moments. The forces and moments are then registered with the vehicle, and the inherited behavior to perform integrations is invoked. All of the complexity associated with integrating the state vector is hidden from the developer.

A high-level representation of several of the major components of the F/A-18 E/F Drop Model is shown in Figure 2. This class diagram shows the use of aggregation and inheritance from the framework. The aggregation would be verbalized as "the F/A-18 E/F Drop Model has an F/A-18 E/F Drop Model Sensor System", etc. The solid black diamonds in Figure 2 indicate that the F/A-18 E/F Drop Model class not only contains these classes but is responsible for their creation and destruction. The use of aggregation provides another means to logically remove complexity and encapsulate data while modeling a system. In this example, the inheritance from the framework establishes a common interface to be used by the sub-classes. It is then the responsibility of an F/A-18 E/F Drop Model Simulation developer to provide these details, as they could not be known at the framework level.

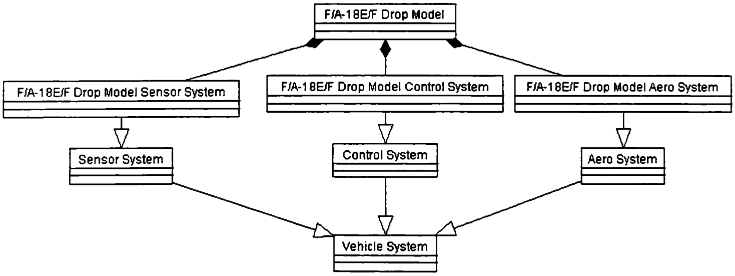

F/A-18E/F Drop Model
F/A-18E/F Drop Model Sensor System
F/A-18E/F Drop Model Control System
F/A-18E/F Drop Model Aero System
Sensor System
Control System
Aero System
Vehicle System

Figure 2: High-Level View

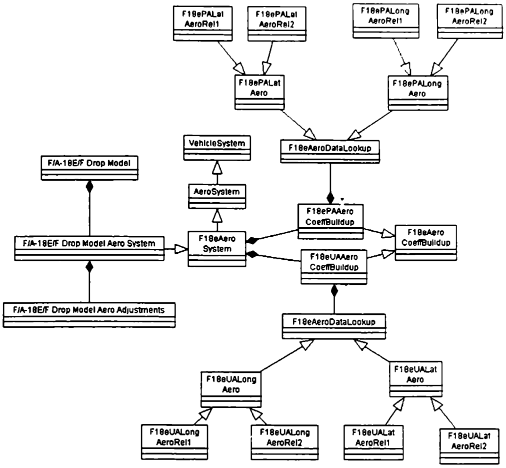


Figure 3: Essentials of the Aerodynamic Model Design

An Object-Oriented Aerodynamic Model

The simulation's aerodynamic model uses two separate databases. One is used for an "up and away" flight regime and the other for a "powered approach" flight regime. Thus, it is necessary to switch between the two databases during synchronous real-time. A class diagram with the details of the aerodynamic model is shown in Figure 3.

Figure 3 shows that the F/A-18 E/F Drop Model has an F/A-18 E/F Drop Model Aero System, which is an F/A-18 E/F Aero System. The only aspects not inherited directly from the full-scale aircraft's aerodynamic model is F/A-18 E/F Drop Model Aero Adjustments. These adjustments represent the differences between the aerodynamic model and the performance of the model being tested. The F/A-18 E/F Aero System has two aero data lookup objects ("F18ePAAeroCoeffBuildup" and "F18eUAAeroCoeffBuildup"). The F/A-18 E/F Aero System acts as a "mediator" between the vehicle and these aerodynamic coefficient build-up objects. The F/A-18 E/F Aero System gets data from the vehicle and passes it on to the appropriate coefficient build-up object. instructs that object to perform its coefficient build-ups and obtains the resulting force and moment coefficients.

These two aerodynamic coefficient build-up classes contain the details of the aerodynamic model. The summation of the aerodynamic coefficients which contribute to final force and moment coefficients is defined therein. The calculation of these coefficients is performed within the aerodynamic data lookup objects that each of the aero coefficient build-up objects contain. The details involved in performing the aerodynamic lookups are grouped into logical sub-classes (e.g. longitudinal and lateral-directional coefficients). The only complexity remaining is the actual data. These classes, diagramed in Figure 3, are represented with a release number (Rel1, Rel2). Subsequent updates to the aerodynamic data result in the creation of a new class with the appropriate release number.

The primary goals of this architecture were to increase maintainability, extensibility and overall reliability. It takes no stretch of the imagination to realize that the aerodynamic model of an aircraft undergoing flight testing will be subject to frequent updates. New releases of the aerodynamic model quite often are limited in scope. At best, only the data in one of the leaf classes (the most specific classes in an inheritance hierarchy) would be changed. The only changes that need to be made to the software are limited to the creation of a class representing the new release. There are no monolithic data files to contend with. The source of any errors that do occur must reside in the new class. The use of aggregation in the design allows the model to be easily extended. Aerodynamic models for any new aspects of the aircraft could be easily handled as a new aggregate of a aerodynamic coefficient build-up class. The interfaces would remain the same and there would simply be another object to service. Through the use of inheritance and aggregation, complexity is dealt with in layers and as smaller, more manageable components which greatly simplifies any aspect of the model.

The Mediator Design Pattern

There is much commonality in the way certain objects interact. These patterns are repeated over and over regardless of the system being modeled. Establishing efficient and effective mechanisms for these relationships is a problem which requires careful attention. Design patterns are evolved, yet simple solutions to these common problems in software development.[2] By using "tried and true" design patterns, a software developer is spared the time and expense of iterating to an efficient solution to the problem.

Figure 4 illustrates the interdependencies that could result when components of a flight control system share data through direct interaction. Each class is then dependent on all the other classes that it needs data from. This creates a tightly coupled system. Any new data or behavior added to one class affects all the other classes that depend on it. As a tightly coupled system grows, it tends to take on the characteristics of a monolithic class. This limits re-usability, hampers testability and significantly increases the time to compile the software. The solution to this common design problem is the Mediator Design Pattern.

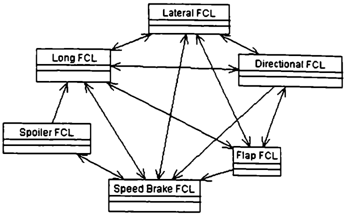


Figure 4: Class Dependencies Abound

The use of the Mediator Design Pattern rids objects of their explicit dependencies. This greatly decouples and simplifies a system. Figure 5 illustrates the impact of a mediator on the system shown in Figure 4.

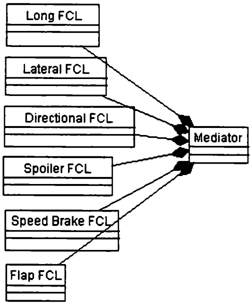


Figure 5: The Mediator Pattern

Figure 5 is the more simple design. It is the superior design. The aggregate classes no longer depend on each other. In fact, they do not even depend on the mediator class which encapsulates them. This autonomy greatly increases re-usability. Testability is increased in that each class may be tested as a single unit, rather than testing the whole system at once. Finally, compilation times are reduced by the fact that a software change that only affects one class will only require re-compilation of that class, not an entire coupled system.

The use of the Mediator Design Pattern in the F/A-18 E/F Drop Model Simulation is shown in Figures 2 and 3. The F/A-18 E/F Drop Model system classes for the aerodynamics, controls and sensors all serve in the role of mediator. These mediator classes take the responsibility for passing data between and managing the behavior of their component objects.

Flight Control Laws and Polymorphism

Figure 6 shows more details of the F/A-18 E/F Drop Model Simulation flight control laws. The features to note are the inheritance relationships from the full-scale F/A-18 E/F classes. The behavior of the F/A-18 E/F Drop Model control laws are for the most part directly inherited from the full-scale implementation. However, there are some aspects of the full-scale control laws that are not used by the drop model.

An example of this can be found in the directional axis control laws. The full-scale laws compute a nose wheel steering command when the control laws are instructed to update. However, such a computation is unnecessary for a drop model, as it has no landing gear. The problem is then to be able to inherit the overall behavior of the control laws, while redefining these kinds of behavioral differences. In a procedural paradigm this would be handled in the baseline code with logical branching. However, as time goes on and other projects use the baseline functionality, this logic would likely evolve into a maintenance nightmare. Within the LaSRS++ framework this sort of behavioral change easily handled thru polymorphism.

The base class (with the baseline set of behaviors) need only define a virtual (the C++ keyword which invokes polymorphic behavior[13, 14]) function representing the behavior. In the sub-class, the same virtual function need only be redefined with the desired behavior. The sub-class can still inherit the fundamental behaviors of the control laws. Only the behavioral aspects that have been polymorphically redefined will change. Once a polymorphic behavior is defined in the base class, it may be easily extended an infinite number of times. Using polymorphism to extend behavior in this manner provides infinite extensibility without creating maintenance burdens. Projects other than the F/A-18 E/F Drop Model could just as easily inherit from the base class and redefine certain behaviors. All sub-classes would be isolated from each other, while at the same time, receiving changes made to the base class.

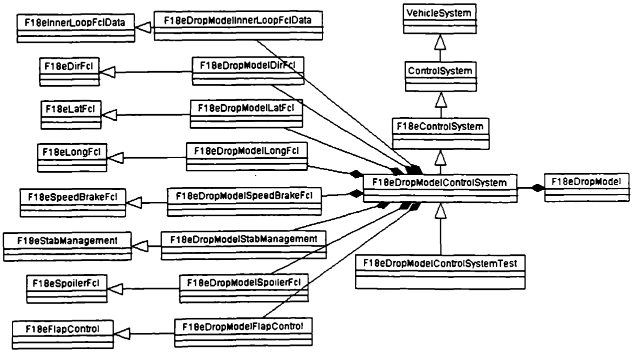


Figure 6: Drop Model Control Law Architecture

Conclusions

A simulation of a 22% dynamically scaled F/A-18 E/F Drop Model was successfully developed using, not only the software, but also the architectural principles of the Langley Standard Real-Time Simulation in C++ (LaSRS++) framework. This simulation's architecture makes frequent use of inheritance, aggregation, polymorphism and the mediator design pattern. This architecture provides strong encapsulation of data while promoting maintainability, extensibility, reliability and re-usability. The high integration rates achieved by the F/A-18 E/F Drop Model Simulation demonstrate the computational efficiency of the LaSRS++ framework.

Acknowledgments

The authors would like to thank Dawn Idema, Stacie Clark and Gus Taylor (Unisys Corporation) for their significant design and implementation contributions to the development of the F/A-18 E/F Drop Model Simulation. Also, Mark A. Croom (Vehicle Dynamics Branch, NASA Langley Research Center) for support, guidance and many technical discussions relating to drop model testing and flight dynamics.

Bibliography

[1] Grady Booch. *Object-Oriented Analysis and Design.* Benjamin/Cummings, Redwood City, California, 1994.

[2] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, Massachusetts, 1995.

[3] John Lakos. *Large-Scale C++ Software Design.* Addison-Wesley, Reading, Massachusetts, 1996.

[4] Robert C. Martin. *Designing Object-Oriented C++ Applications Using The Booch Method.* Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[5] Mark A. Croom, et al. Dynamic Model Testing of the X-31 Configuration for High-Angle-of-Attack Flight Dynamics Research. Paper Number AIAA-93-3674 CP, August, 1993.

[6] C. H. Wolowicz, J. S. Bowman, W. P. Gilbert. Similitude Requirements and Scaling Relationships as Applied to Model Testing. Technical Report NASA TP 1438, 1979.

[7] Richard A. Leslie, et al. LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft. Paper Number AIAA-98-4529, August, 1998.

[8] David Geyer, et al. Managing Shared Memory Spaces in an Object-Oriented Real-time Simulation. Paper Number AIAA-98-4532, August, 1998.

[9] Patricia Glaab, et al. A Method to Interface Auto-Generated Code into an Object-Oriented Simulation. Paper Number AIAA-98-4531, August, 1998.

[10] Michael Madden, et al. Constructing a Multiple-Vehicle, Multiple-CPU Using Object-Oriented C++. Paper Number AIAA-98-4530, August, 1998.

[11] P. Sean Kenney, et al. Using Abstraction to Isolate Hardware in an Object-Oriented Simulation. Paper Number AIAA-98-4533, August, 1998.

[12] Terry Quatrani. *Visual Modeling With Rational Rose and UML.* Addison Wesley, Reading, Massachusetts, 1998.

[13] Bruce Eckel. *Thinking in C++.* Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[14] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley Publishing Company, Reading, Massachusetts, third edition, 1997.

[15] Scott Meyers. *Effective C++.* Addison-Wesley, Reading, Massachusetts, second edition, 1998.

[16] Scott Meyers. *More Effective C++.* Addison-Wesley, Reading, Massachusetts, 1996.

[17] Atul Saini David R. Musser. *STL Tutorial and Reference Guid.* Addison-Wesley, Reading, Massachusetts, 1996.

AIAA-98-4161

SIMULATING CONCEPTUAL AND DEVELOPMENTAL AIRCRAFT

Joseph J. Totah and Dr. David J. Kinney
Aerospace Engineers
NASA Ames Research Center
Moffett Field, California

Abstract

This paper presents results of a new capability to perform real-time simulation of conceptual and developmental aircraft. The objective is to seamlessly perform real-time simulation of arbitrary aircraft configurations whose geometric, inertial, and aerodynamic characteristics are either specified or estimated using conceptual design software, and then flown with a pre-existing flight control architecture that does not require gain scheduling or redesign. This new capability is first examined by comparing design estimates with known characteristics of an existing aircraft. The results correlate well with known inertial and closed-loop dynamic characteristics, however limitations are noted in the estimates of the aerodynamics. Another aircraft examined is that of a concept designed to fly on the surface of Mars. Although correlation data does not exist for this aircraft, the results indicated the conceptual Mars aircraft exhibits well behaved closed-loop dynamic characteristics with some coupling noted in the directional axis that may be attributed to spiral instability. These results represent a first step towards the completion of an integrated tool to simulate conceptual and developmental aircraft.

Introduction

This paper examines a value-added enhancement to the aircraft synthesis process for new and existing aircraft. This enhancement is the development of an interactive flight simulation capability for conceptual aircraft, wind tunnel models, existing aircraft, and other aerospace vehicles designed for piloted, remote, or autonomous operations. The approach is to provide a pilot or ground-based operator the capability to fly or control an arbitrary design in a simulated environment. It allows pilot or operator feedback to enhance the aerospace research and development process very early in the aircraft's life-cycle, as well as improve the relevance of preliminary designs well into the final design, pre-production, and production stages of development.

The later stages of the aerospace research and development process often include wind tunnel testing and flight experimentation, which can result in tremendous cost overruns if design changes are required to improve vehicle flying qualities, account for additional payloads, or accommodate growth to meet expanding mission requirements. Among the most expensive and time consuming stages of this process are the early prototype testing and the development of the flight control software. For example, industry figures for one particular family of production aircraft averaged over 1.7 million developer hours, 100 software versions, and 500 test flights to develop the current flight control software.

In fact, increasingly high cost, increased foreign competition, and slow development times associated with new aircraft designs have lead to a major industry objective of reducing new flight control software and design specification processes by 50% in five years.

A key technology that makes it possible to rapidly simulate conceptual and developmental aircraft is neural adaptive flight controls. These types of controls can be adapted quite easily to a wide class of aerospace vehicles including commercial transports, high performance military aircraft, hypersonic vehicles, remotely piloted concepts, rotorcraft, reusable launch vehicles, and autonomous planetary aircraft. It is believed that a 157 to 1 time advantage and a $0.5M cost savings for each flight control system prototype can be realized using these types of controls.

In fact, direct adaptive tracking control is ideally suited for this application given the geometry, mass and inertia characteristics, and aerodynamics of the vehicle under consideration. Gain scheduling is not required with this controller because the architecture is based on feedback linearization theory. A tuning capability allows specification of vehicle class, flight phase category, and level of flying qualities by simply varying inner loop and outer loop time constants, both of which are functions of natural frequency and damping of the closed-loop transfer function of an equivalent second order system. Neural networks are used to compress and

Senior Members

accelerate the mapping and recall of the aerodynamic data used in the inverse model portion of the controller.

The work presented in this paper was funded by the Information Technology R&T Base Program managed at NASA Ames Research Center. Contributing elements of that program are the Analytical Tools and Environments for Design (ATED) and Intelligent Systems Control and Operations (ISCO), whose missions are to leverage information technologies and core competencies in soft computing and computational intelligence to support NASA's Office of Aeronautics and Space Transportation Technology Enterprise in achieving broader Agency goals over the next five years. This is a cooperative effort between the Computational Sciences Division and the Aeronautical Project and Programs Office at NASA Ames Research Center.

Design Environment

Several steps are performed in order to simulate conceptual and developmental aircraft. These steps comprise the "Design Environment", shown in figure 1.

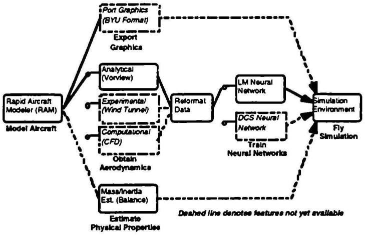


Figure 1. Design Environment.

The first step in the Design Environment is to generate the aircraft's geometric shape. Currently, the software used to generate the three-dimensional image of the aircraft is the Rapid Aircraft Modeler (RAM), developed internally at NASA Ames. Two completely different types of aircraft were drawn using RAM; the F-15 Advanced Control Technologies for Integrated Vehicles (ACTIVE) shown in figure 2, and a conceptual Mars aircraft shown in figure 3. The F-15 ACTIVE aircraft is currently in operation at NASA Dryden, considerable data exist for it. On the other hand, the only data specified for the conceptual Mars aircraft are wing span, wing area, aspect ratio, component masses, and design cruise speed at the surface of Mars.

The F-15 ACTIVE RAM model was completed in approximately one week. It is a complicated model of an actual aircraft currently in operation at NASA Dryden. Care was taken to maintain as much geometric integrity as possible. The conceptual Mars aircraft was completed in one day. It is a much less complicated model with fewer and less detailed components.

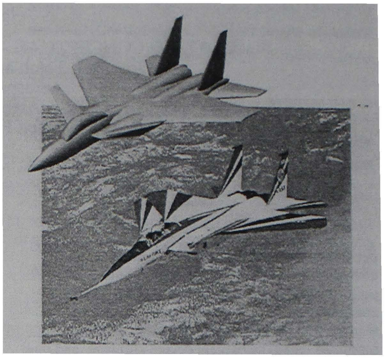


Figure 2. F-15 ACTIVE RAM Model.

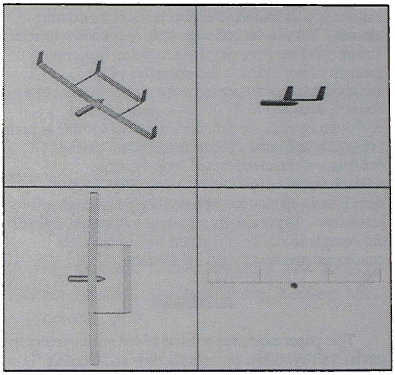


Figure 3. Conceptual Mars Aircraft RAM Model.

Once the RAM models are generated, the code "Balance" is then used to estimate aircraft center-of-gravity (CG) and inertial characteristics. This code was also developed internally at NASA Ames, and it has sufficient flexibility to specify internal and external component weights and locations. For configurations with known internal or external component weights and locations, those weights are first subtracted from the total vehicle weight and the difference is then uniformly distributed over the remaining volume. The outputs of interest are the mass moments of inertia and center-of-gravity location relative to the nose of the aircraft, I_x, I_y, I_z, I_{xz}, X_{CG}, and Z_{CG}.

There are many ways to estimate or measure vehicle aerodynamics, such as semi-empirical expressions, theoretical aerodynamics, wind tunnel experimentation, computational fluid dynamics, flight experimentation, or any mixture thereof. The aerodynamic tool currently available within the Design Environment is the vortex-lattice code Vorview[1], which was recently modified to generate stability and control derivatives for specified operating conditions. Vorview outputs an ASCII text file and also displays a colormap image of the surface pressure distribution on the monitor. A grayscale example of the colormap is shown in figure 4 for the conceptual Mars aircraft.

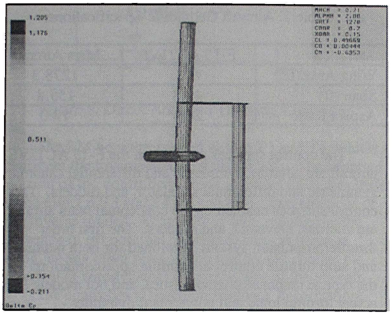


Figure 4. Grayscale Mapping of the Surface Pressure Distribution for the Conceptual Mars Aircraft.

Vorview is executed repeatedly at a variety of operating conditions that define the flight envelope of interest. All of the ASCII text files generated by Vorview are reformatted into columns of data within one single file. The first column(s) are independent variables (typically Mach number, angle-of-attack, etc...), and the remaining columns are dependent variables, which are the stability and control derivatives.

This reformatted data is then used to train a neural network. A neural network is used instead of the more traditional approach of using table look-ups because real-time simulation performance is enhanced in terms of reduced size, accelerated recall, improved interpolating accuracy, and graceful degradation when extrapolating outside the operational flight envelope.[4]

Options available for training include a "Levenberg-Marquardt" (LM) multi-layer perceptron[2], and a "Dynamic Cell Structure" (DCS) perfect topology representation network[3], both of which can be trained to predetermined error levels. The trained network is used in both the equations of motion and the inverse model portion of the flight control architecture, described in the next section.

Simulation Environment

Once the three-dimensional RAM model of the aircraft is developed it can be exported into the "Simulation Environment" that includes a graphical database and texture mapped terrain. A block diagram of the Simulation Environment is shown in figure 5.

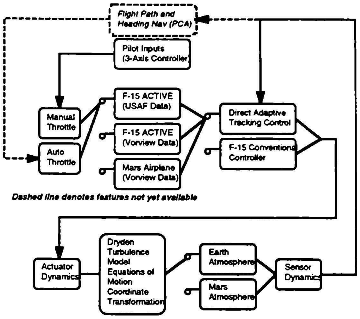

Flight Path and Heading Nav (PCA)
Pilot Inputs (3-Axis Controller)
Manual Throttle
Auto Throttle
F-15 ACTIVE (USAF Data)
F-15 ACTIVE (Vorview Data)
Mars Airplane (Vorview Data)
Direct Adaptive Tracking Control
F-15 Conventional Controller
Dashed line denotes features not yet available
Actuator Dynamics
Dryden Turbulence Model Equations of Motion Coordinate Transformation
Earth Atmosphere
Mars Atmosphere
Sensor Dynamics

Figure 5. Simulation Environment.

The math model that drives the graphics is complete with sensor and actuator dynamics and a first order throttle/propulsion system with both manual and auto throttle control. The equations of motion are based on perturbation theory, however most of the non-linear terms are retained in the gravitational and inertial portions of the translational axes. All of the non-linear terms are retained in the inertial portions of the rotational axes.

Environmental conditions for both Earth and Mars are modeled to account for pressure, density, temperature, gravity, and speed-of-sound variations with altitude; from mean sea level up to 282,166 feet (86 km) for Earth and 196,860 feet (60 km) for Mars. Aerodynamic derivatives, conventional control system gains, and nominal operating conditions are scheduled to allow full flight envelope simulation at altitudes up to the design limits of each aircraft.

The Earth atmosphere is based on a 1976 standard atmosphere model. The Dryden turbulence model is implemented in a separate subroutine to provide an option for simulating light, medium, or heavy turbulence levels at altitudes below 10,000 feet.

The Mars atmosphere is based on a Global Reference Atmosphere Model (GRAM), which, among other things, requires a specific reference date and trajectory.[5] The model includes crosswind variations relative to isobars (lines of constant pressure), which vary in direction and magnitude relative to altitude. The date and trajectory were chosen to be those of the recent Pathfinder mission; July 4, 1997 at 19.3 degrees North latitude and 33.6 degrees West longitude. Plots of the atmospheric parameters for Mars on that date and trajectory calculated by GRAM are shown in figure 6.

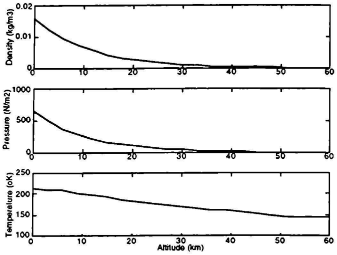


Figure 6. Mars GRAM Atmospheric Model Estimates.

The simulation is integrated with a high-fidelity graphics database, a simulator pod with visual and auditory feedback, a three-axis hand controller, throttle lever, and push-button switches to provide an excellent, low-cost fixed-base simulator for pilot-in-the-loop simulation, shown in figure 7. Texture-mapped terrains were developed for both Earth and Mars to provide high quality visual cueing. The Earth terrain is complete with a runway, mountain ranges, and various sized hangars. The Mars terrain is modeled after Valles Marineris, which was downloaded from the internet and enhanced for incorporation into the Simulation Environment.

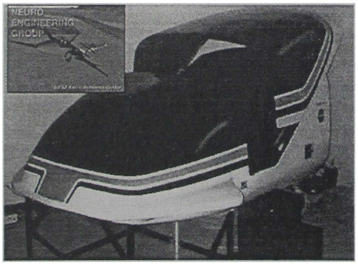


Figure 7. Simulator Pod and Graphics.

Aircraft math models were developed for both the F-15 ACTIVE and the conceptual Mars aircraft. The F-15 ACTIVE is modeled after configuration G of the US Air Force's Short takeoff and Landing Maneuver Technology Demonstrator (S/MTD) program.[6] The conceptual Mars aircraft is modeled after one of six candidate configurations designed for a long endurance mission on Mars.[7] Two of the six configurations are radioisotope powered and four are solar powered, all with different geometry's to accommodate various powerplant options. The Mars concept considered herein is one of the solar powered configurations using gallium arsenide solar cells. Both the F-15 ACTIVE and the conceptual Mars aircraft assume gear/flaps retracted, no external payloads, and no thrust vectoring capability. Geometric specifications are given in Table 1 for both aircraft.

Table 1. Aircraft Geometric Specifications.

	F-15 ACTIVE	Mars Aircraft
Wing Area (ft^2)	608	1278.3
Span (ft) .	42.8	155.8
Aspect Ratio	2.69	19.0

The control surfaces enabled on the F-15 ACTIVE aircraft are ailerons, symmetric and differential canards, symmetric and differential stabilator, and rudders. The control surfaces enabled on the conceptual mars aircraft are ailerons, elevators, and rudders. The first order throttle/propulsion system developed for both manual and auto throttle control are simple approximations of the typical response characteristics, and not models of actual throttle logic and powerplant dynamics.

The direct adaptive tracking controller developed by Kim and Calise was implemented by NASA with several modifications.[8,9] The two most important modifications are that the pre-trained and on-line learning radial basis neural networks in the original design were replaced by a LM and DCS neural network, respectively. These networks were developed at NASA Ames, and they meet several performance metrics relative to accuracy, speed, size, locality, and stability.

The overall architecture is based on feedback linearization theory, and is comprised of a command augmentation system (CAS) that has both an attitude orientation system and command augmentation logic. The attitude orientation system is an airframe stabilization system that accepts rate commands. This serves the function of stabilizing the airframe while following the commanded rates. The command augmentation logic is an outer loop function to track the pilot/operator commands. The pilot/operator commands normal acceleration with longitudinal stick, lateral acceleration with pedals, and roll rate with lateral stick. This design provides normal acceleration and roll rate command tracking and automatic turn coordination.

The output of the attitude orientation system (U_{1pd}, U_{2pd}, and U_{3pd}) is summed with the adaptive portion of the controller (U_{1ad}, U_{2ad}, and U_{3ad}) to generate U_1, U_2, and U_3, which are then transformed into commanded aircraft body axis rotational accelerations (T3). These accelerations are the inputs to the inverted model (T4) for calculating control surface deflections, as shown in figure 8.

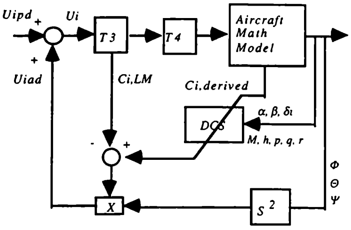


Figure 8. Direct Adaptive Tracking Controller.

The LM network is used within T3 and T4, which represent the inverse model of the aircraft. The adaptive portion of the controller, which includes DCS, has little or no effect if T3 and T4 are accurate representations of the inverse model. Adaptivity is only required in the presence of off nominal conditions such as damage, icing, or any other phenomena that would change the physical or aerodynamic characteristics of the aircraft.

The controller is based on feedback linearization theory, and no gain scheduling is required for the architecture. The gains are functions of stability and control derivatives that can be tuned to produce desirable natural frequency, w_n, and damping, ς, characteristics of an equivalent, complex, linear second order system. Tuning is accomplished by varying the inner-loop and outer-loop time constants which, in turn, affect gains within the CAS. The following expressions are used to tune the controller:

$$t_{s_i} = \frac{4.6}{\varsigma w_n} \quad (1)$$

$$t_{s_o} = 3t_{s_i} \quad (2)$$

$$K_d = \frac{9.2}{t_{s_i}} = 2\varsigma w_n \quad (3)$$

$$K_p = \frac{K_d^2}{2} = w_n^2 \quad (4)$$

$$\tau_n = \frac{2m}{\rho S V C_{L_\alpha}} \quad (5)$$

$$\tau_y = \frac{2m}{\rho S V \left| C_{Y_\beta} \right|} \quad (6)$$

$$K_2 = K_4 = \frac{4.6}{t_{s_o}} \quad (7)$$

$$K_1 = \tau_n K_2 \quad (8)$$

$$K_3 = \tau_y K_4 \quad (9)$$

Within the CAS, K_d and K_p comprise part of the attitude orientation system that form U_{1pd}, U_{2pd}, and U_{3pd}. K_1, K_2, K_3, and K_4 comprise part of the command augmentation logic used to track pilot/operator commands. Thus, tuning of the CAS is accomplished simply by varying t_{s_i} and t_{s_o}, which, when properly specified, result in defining the vehicle class, flight phase category, and level of flying qualities for a given aircraft.

Results

The results presented in this section provide a basis with which to establish the abilities of the both the Design and Simulation Environments to model and simulate conceptual and developmental aircraft. These results are primarily influenced by the capabilities of RAM, Vorview, and Balance to accurately estimate the physical and aerodynamic characteristics of interest. They are also dependent on the ability of the controller to produce acceptable closed-loop dynamic characteristics of the aircraft.

Overall, much attention was spent in developing the F-15 ACTIVE RAM model. Known physical and aerodynamic characteristics were used to validate the results produced by Balance and Vorview. The LM neural network was trained on the reformatted Vorview estimates, and then used along with the Balance estimates in the Simulation Environment to examine the ability of the direct adaptive tracking control architecture to match the closed-loop dynamic characteristics of the aircraft. The process was then repeated for the conceptual Mars aircraft.

Validation for the F-15 ACTIVE aircraft was performed by matching perturbed state time histories at various flight conditions to those generated by the full, non-linear, six-degree-of-freedom simulation conducted by the Air Force during the S/MTD program. Validation was not performed for the conceptual Mars aircraft because of the absence of correlation data. For both aircraft, automatic turn coordination was not used.

The maneuver chosen to illustrate the ability of the Simulation Environment to examine the F-15 ACTIVE response characteristics is as follows: starting at M_o =

0.4 and h_o = 10,000 ft, a roll rate step command of 10 deg/sec is given until an 80 deg bank angle is obtained. Normal acceleration is maintained at 1.0g during the roll maneuver. This is followed by a 5.0g normal acceleration step command that is maintained until the end of the maneuver. The throttle is held at 100%, resulting in a speed of M = 1.0 after 30 seconds.

Figure 9 shows the predicted response characteristics of the data estimated by Balance and Vorview when integrated into the direct adaptive tracking control architecture within the Simulation Environment. The short period dynamics for roll rate are well predicted in terms of rise time, but exhibit noticeable overshoot at the onset and completion of the step command. The desired command is captured and maintained after each overshoot.

The opposite is seen with normal acceleration, such that predicted rise time is slower in achieving the commanded input, and does not exhibit the overshoot produced by the actual aircraft. It is believed that the overshoot is the result of a $-\frac{g}{v_t}(1-\cos\theta\cos\phi)$ term added to pitch rate in the control laws of the actual aircraft, which tend to improve responsiveness but decrease damping at large bank or pitch angles. The desired command is ultimately captured and maintained, however a departure is noted at speeds approaching M = 1.0, where Vorview estimates become unreliable.

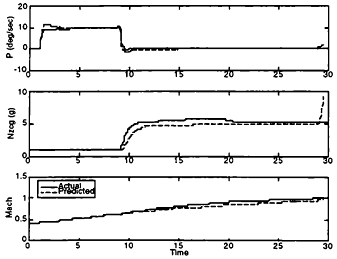


Figure 9. F-15 ACTIVE Predicted Maneuver Response.

The conceptual Mars aircraft was examined only at the design cruise speed and altitude. Angular rate responses to longitudinal, lateral, and directional axis doublets are shown in figure 10. The behavior appears to be well behaved in pitch and roll, with coupling noted in the directional axis. This coupling, when flown interactively, was noted to be objectionable. It was discovered that the estimated value of $C_{L_o}\left(C_{l_\beta}C_{n_r} - C_{n_\beta}C_{l_r}\right)$ is slightly negative, which can be a characteristic of spiral instability depending on the lateral stability boundary for that aircraft.[10]

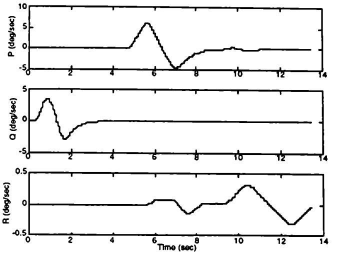


Figure 10. Conceptual Mars Aircraft Predicted Response to Stick and Pedal Doublets (M = 0.2, h = 100 ft).

The results presented in Table 2 illustrate the accuracy of Balance in estimating F-15 ACTIVE CG location and inertia values, all of which are acceptable relative to actual values for that aircraft. Table 3 shows the Balance estimates for the conceptual Mars aircraft.

Table 2. F-15 ACTIVE Balance Accuracy's.

Moment of Inertia	Error (Balance vs. Actual)
I_x	-1.4%
I_y	-0.6%
I_z	2.1%
I_{xz} *	-141.7%
X_{CG}	3.1%
Z_{CG} *	49.3%

* The actual values of I_{xz} and Z_{CG} are very small.

It should be noted that for the F-15 ACTIVE aircraft, differences in estimated versus actual values for I_{xz} and Z_{CG} are not that great, and are on the order of the differences for I_{ZZ} and X_{CG}, respectively. The percent errors for those parameters are large because their actual values are relatively small to begin with.

Table 3. Conceptual Mars Aircraft Balance Estimates.

Moment of Inertia	Balance Estimates
I_x	661878.9 (lb-ft2)
I_y	136977.6 (lb-ft2)
I_z	771161.1 (lb-ft2)
I_{xz}	-8920.9 (lb-ft2)
X_{CG}	21.5 ft
Z_{CG}	2.2 ft

The F-15 ACTIVE internal component specification is fuel distribution, and external component specifications are the canards and the nacelle/engine combination. The conceptual Mars aircraft has no internal component specifications (it is a solar powered concept with no fuel distribution), and external component specifications are the winglets, tailbooms, nacelle/engine, and propeller.

The results presented in Table 4 illustrate the accuracy of Vorview in estimating relatively important stability derivatives for the F-15 ACTIVE aircraft.[11] The results are very good for C_{m_α} and C_{m_q}, marginal for C_{l_β} and C_{l_p}, and poor for C_{Z_α} and C_{n_β} when considered in the context of similar methods or semi-empirical expressions. Runs were performed for Mach numbers ranging between 0.3 and 1.3, however accuracy's are not shown in the transonic region because the results degrade significantly at those higher speeds. These results are consistent with limitations of vortex-lattice methods, in general. The utility of Vorview is realized in terms of ease-of-use and computational efficiency, in the absence of more reliable data generated from other experimental or computational sources.

Table 4. F-15 ACTIVE Vorview Accuracy's.

M	C_{Z_α} error (%)	C_{m_α} error (%)	C_{m_q} error (%)	C_{n_β} error (%)	C_{l_β} error (%)	C_{l_p} error (%)
0.3	25.6	0.9	12.7	5.0	-39.8	1.6
0.4	19.9	-0.8	10.4	60.0	-19.3	-13.8
0.5	20.4	-5.0	8.2	59.2	-16.6	-20.5
0.6	20.8	3.0	5.2	55.7	-15.6	-27.8
0.7	21.0	-2.3	0.6	55.7	-12.1	-36.1
0.8	20.9	-8.9	-6.0	52.0	-14.7	-54.3
|Avg|	21.5	3.5	7.2	47.9	19.7	25.7

Table 5 shows derivative values for the conceptual Mars aircraft at the design cruise speed of $M = 0.2$. The speed-of-sound and gravitational acceleration on the surface of Mars are approximately 2.5 times less than the corresponding sea-level values on Earth.

Table 5. Conceptual Mars Aircraft Vorview Estimates.

M	C_{Z_α} 1/rad	C_{m_α} 1/rad	C_{m_q} 1/rad	C_{n_β} 1/rad	C_{l_β} 1/rad	C_{l_p} 1/rad
0.20	-6.97	-7.09	-60.8	.0165	-.036	-.803

Once the F-15 ACTIVE derivatives are generated and reformatted, an LM neural network is trained and tested on the entire data set. The LM network is a multi-layered perceptron with six processing elements in a single hidden layer. The activation function used in the hidden layer is of the form $1 - e^{-x^2}$. The trained network was used in both the equations of motion and the inverse model portion of the flight control architecture to enhance real-time simulation performance. A comparison of neural networks and table look-ups is provided in Table 6 to illustrate some performance metrics of interest.

Table 6. Comparison of Table Look-up and Neural Network Performance in a Real-Time Simulation.

Metrics	Size (bytes)	Speed (sec)	RMS Error
Tables	101472	0.31	0%
Neural Net	7175	0.16	< ±3.5%
Performance	14:1	2:1*	Acceptable

* Cumulative execution time for 1 million iterations.

Training was not performed on the derivatives estimated by Vorview for the conceptual Mars aircraft because data were only generated at a single airspeed (the design cruise speed) for that configuration. The total amount of data did not comprise a large enough training set to realize any of the aforementioned benefits.

Conclusion

The results presented in the previous section demonstrate the foundation of a new capability to perform real-time simulation of conceptual and developmental aircraft. A Design Environment and Simulation Environment are developed such that arbitrary aircraft configurations whose geometric, inertial, and aerodynamic characteristics are either specified or estimated using conceptual design software, and then flown with a pre-existing flight control architecture that does not require gain scheduling or redesign. This new capability is first examined by comparing design estimates with known characteristics of an existing aircraft; namely the F-15 ACTIVE aircraft. The results are correlated with respect to modeling the F-15 ACTIVE geometry, estimating the aircraft's inertial characteristics, estimating stability and control derivatives throughout a wide range of speeds, and matching closed-loop dynamic characteristics. This correlation provides the necessary basis with which to assess the validity of the conceptual design software and the flight control architecture in simulating entirely new concepts.

One such concept examined is that of an aircraft designed to fly on the surface of Mars. Again, aircraft geometry is modeled and physical and aerodynamic characteristics are estimated using the conceptual design software. Results indicate the conceptual Mars aircraft exhibits well behaved closed-loop dynamic characteristics in the longitudinal and lateral axes, with some coupling noted in the directional axis that is shown to be attributed to spiral instability.

Although the results are considered preliminary, they represent a major step towards the completion of an integrated tool to simulate conceptual and developmental aircraft very early in the aerospace research and development cycle.

References

[1] Miranda, L. R., Elliot, R. D., and Baker, W. M., "A Generalized Vortex Lattice Method for Subsonic and Supersonic Flow Applications", NASA CR 2865, December, 1977.

[2] Norgaard, M., Jorgensen, C., and Ross, J., "Neural Network Prediction of New Aircraft Design Coefficients", NASA TM 112197, May 1997.

[3] Jorgensen, C. C., "Direct Adaptive Aircraft Control Using Neural Networks", NASA TM 47136, January, 1997.

[4] Totah, J., "An Examination of Aircraft Aerodynamic Estimation Using Neural Networks", SAE Paper 952036, 1995.

[5] Justus, C. G., Johnson, D. L., and James, B. F. "A Revised Thermosphere for the Mars Global Reference Atmospheric Model (Mars-GRAM Version 3.4)", NASA TM 10851, July 1996.

[6] Zeh, J. M., Young, D. L., and Couture, N. J., "STOL and Maneuver Technology Demonstrator (S/MTD) Stability Derivatives", WL-TR-92-3055, 1992.

[7] Colozza, A. J., "Preliminary Design of a Long-Endurance Mars Aircraft", NASA CR 185243, April 1990.

[8] Kim, B. S. and Calise, A. J., "Nonlinear Flight Control Using Neural Networks", AIAA Journal of Guidance, Navigation, and Control, Vol. 20, No. 1, 1997.

[9] Totah, J., "Adaptive Flight Control and On-Line Learning", AIAA Paper 97-3537, 1997.

[10] Etkin B., Dynamics of Flight - Stability and Control , John Wiley and Sons, New York, 1982.

[11] Roskam, J., Airplane Flight Dynamics and Automatic Flight Controls, Part I, Roskam Aviation and Engineering Corporation, Ottawa, Kansas, 1979.

AIAA-98-4162

FUZZY LOGIC BASED FUNCTION MODELING IN AERODYNAMIC SIMULATIONS: IMPLEMENTATION AND TIMING

Douglas D. Hensley*
R&D Enterprises
359 30th Avenue
Newton, Kansas 67114
E-Mail: dhensley@midusa.net

Abstract

This paper is part of a continuing investigation into the use of fuzzy logic based function modeling as an alternative to linear table look-up and interpolation in real-time flight simulation. It presents a discussion of implementation issues that need to be addressed when considering a fuzzy logic based function model for a real-time aerodynamic simulation. These issues consist of the method of representing the input and output fuzzy membership sets and the choice of inferencing method. A timing comparison between the various fuzzy logic systems demonstrating the effect of the above implementation issues on computational performance will be presented. A timing comparison between these fuzzy based systems and a linear table look-up and interpolation algorithm will also be presented.

Three methods of defining the set membership were considered: the Centers method, the Center and Radius method, and the Gaussian method. The Gaussian method incurs a significant computational performance penalty compared to the other two methods.

In a previous paper, three inferencing methods were presented as appropriate choices for accurate function modeling in a time critical environment: the Sum-Product, Sum-Mean and Time-Critical Optimized inferencing methods. Of these methods, the Sum-Mean inferencing method requires about ten percent greater computational resources. In this paper, a newly emerging inferencing method called the Combs Union Matrix method is introduced. This method proved to be 1.5 and 8 times faster for one- and two-dimensional functions respectively.

In timing comparisons with a linear table look-up and interpolation algorithm, only the Combs Union Matrix method showed a potentially comparable performance. The requirement to calculate numerous membership set values for every function evaluation, appears to be the greatest handicap for real-time applications of fuzzy logic based function modeling.

Introduction

Typically function modeling in real-time flight simulation is performed as a linear table look-up and interpolation in N dimensions, where N represents the number of independent variables used to define the function model. The linear interpolation method uses mathematically simple operators and can be efficiently implemented. Modeling resolution in non-linear regions of the function being modeled is achieved through denser breakpoint spacing along the independent variable's domain. One drawback of the linear table look-up is the 2^N exponential growth in the number of calculations required to perform a multi-parameter interpolation.

Fuzzy logic presents a method of function modeling that deserves consideration as a possible alternate to the linear table look-up and interpolation method. Fuzzy systems have been shown to belong to the class of functions that are capable of approximating any real function to any desired degree of accuracy. These fuzzy systems are built upon very simple and efficient mathematical functions and can be implemented in a variety of forms.

This paper is part of a continuing investigation into the use of fuzzy logic based function modeling as an alternative to linear table look-up and interpolation in real-time flight simulation. In a previous paper, the background for an investigation into the use of fuzzy logic based function modeling was presented. In that paper, three inferencing methods were presented as appropriate choices for accurate function modeling in a time critical environment: the Sum-Product, Sum-Mean and Time-Critical Optimized inferencing methods. Implementing these inferencing methods as fuzzy clustering rules with the output consequent defined as a linear function of the input variables were shown to significantly improve the accuracy, flexibility and size of the resulting function model.

*AIAA Member

This paper presents a discussion of implementation issues that need to be addressed when considering a fuzzy logic based function model for a real-time aerodynamic simulation. These issues consist of the method of representing the input and output fuzzy membership sets and the choice of inferencing method. A newly emerging inferencing method called the Combs Union Matrix method is introduced. A timing comparison between the various fuzzy logic systems demonstrating the effect of the above implementation issues on computational performance will be presented. A timing comparison between these fuzzy based systems and a linear table look-up and interpolation algorithm will also be presented.

Background

Fuzzy Membership Sets

The three most commonly used methods of defining fuzzy membership sets are the Centers Method, the Center and Radius Method, and the Gaussian Method. These membership set definition methods are depicted in Figure 1.

The fuzzy set membership values can be calculated respectively as:

$$\begin{array}{ll} \text{If } x < C_n & \mu(x) = \text{Max}[1 - (C_n - x)/(C_n - C_{n-1}), 0] \\ \text{Else} & \mu(x) = \text{Max}[1 - (x - C_n)/(C_{n+1} - C_n), 0] \end{array} \quad (1)$$

$$\mu(x) = \text{Max}(\, 1 - |C_n - x| \,/\, R_n, 0) \quad (2)$$

$$\mu(x) = e^{-0.5[(C_n - x)/\sigma_n]^2} \quad (3)$$

Fuzzy Inferencing

Relationships between fuzzy sets of several variables can be expressed in terms of IF-THEN rules defining the fuzzy implications of a fuzzy system. These rules would take the general form:[1,2]

If In_1 is A_1 & In_2 is A_2 & ... Then Out_1 is B_1 (4)
Where: A_1, A_2, and B_1 are fuzzy sets.

The rule consequent, B_1 of this general rule form could be assigned a singleton, or constant value. This rule consequent could alternatively be defined as a linear function of the input variables:[3]

If In_1 is A_1 & In_2 is A_2 & ... Then $Out_1 = C_0 + C_1In_1 + C_2In_2 + \ldots$ (5)

The rules shown in Equations 4 and 5 are of the form of the traditional intersection method of fuzzy inferencing. Table 1 shows a typical intersection rule matrix formed by a two input, one output fuzzy system.

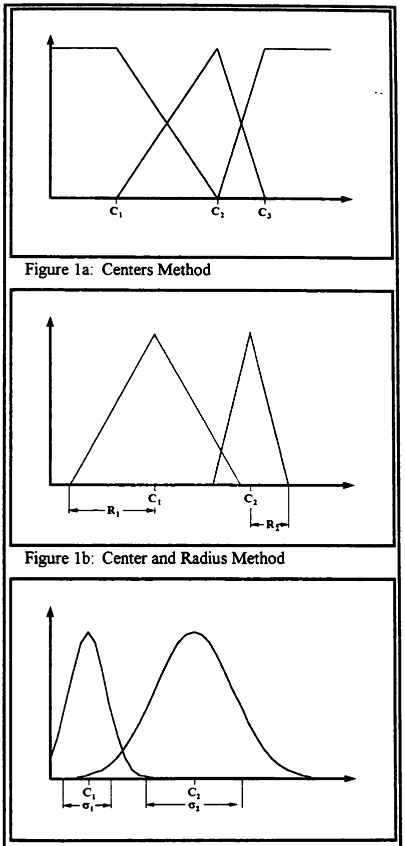


Figure 1a: Centers Method

Figure 1b: Center and Radius Method

Figure 1c: Gaussian Method

Figure 1: Fuzzy Membership Set Representations

A newly emerging fuzzy inferencing method, named the Combs Method, is defined using a union rule matrix rather than the intersection rule matrix. Rules composed by this method would take the form:[4]

	In2 Set1	Set2	Set3
In1 Set1	Out1	Out1	Out2
Set2	Out1	Out2	Out3
Set3	Out2	Out3	Out3

Table 1: Typical Intersection Rule Matrix

If In$_1$ is A$_1$ Then Out$_1$ is B$_1$ (6)
OR
If In$_2$ is A$_2$ Then Out$_1$ is B$_1$
Where: A$_1$, A$_2$, and B$_1$ are fuzzy sets.

An example union rule matrix is shown in Table 2.

In1	Set1	Set2	Set3
	OR		
In2	Set1	Set2	Set3
Out	Out1	Out2	Out3

Table 2: Typical Union Rule Matrix

Non-Classical Inferencing Operators

The choice of fuzzy set operators and resulting inferencing method can significantly affect the characteristics of a fuzzy system. The available number of non-classical operators appropriate for function modeling applications is considerably reduced when applicability within a time-critical environment is a factor.[5] Some operators and inferencing methods that satisfy these conditions would include the Sum-Product[1,2], Sum-Mean[1,2], and Time-Critical Optimized[6] inferencing methods. These inferencing methods are defined respectively as:

$$(\mu_A(x_1) \times \mu_B(x_1)) + (\mu_A(x_2) \times \mu_B(x_2)) \qquad (7)$$

$$(\mu_A(x_1) + \mu_B(x_1)) / 2 + (\mu_A(x_2) + \mu_B(x_2)) / 2 \qquad (8)$$

$$(\mathrm{Min}(\mu_A(x_1), \mu_B(x_1)) \times \mathrm{Mean}(\mu_A(x_1), \mu_B(x_1)) + \mathrm{Min}(\mu_A(x_2), \mu_B(x_2)) \times \mathrm{Mean}(\mu_A(x_2), \mu_B(x_2))) / 2 \qquad (9)$$

Fuzzy Function Modeling

In general, a fuzzy system consists of a set of rules defining relationships between the input state space and the output state space. These fuzzy rules form fuzzy patches that can be used for function approximation. Fuzzy systems approximate functions by covering their graphs with fuzzy patches; the accuracy of the function approximation increases as the number of fuzzy patches increases and the size of the patches decreases. Fuzzy systems have been demonstrated to belong to the class of functions which are capable of approximating any real function to any desired degree of accuracy.[7]

Function Modeling: Fuzzy Associative Memory

The intersection matrix rule forms of Equations 4 and 5, quantize the input-output state space into a Cartesian product, X×Y, of potential fuzzy rule combinations. These potential fuzzy rules form fuzzy patches that can be used for function approximation. The potential fuzzy rule combinations formed by the quantization of the input-output state space is called the Fuzzy Associative Memory (FAM).[8] The FAM method is currently the most common method used to implement fuzzy systems.

Function Modeling: Fuzzy Clustering

In the fuzzy clustering approach to function modeling, the potential fuzzy rule combinations are not based upon

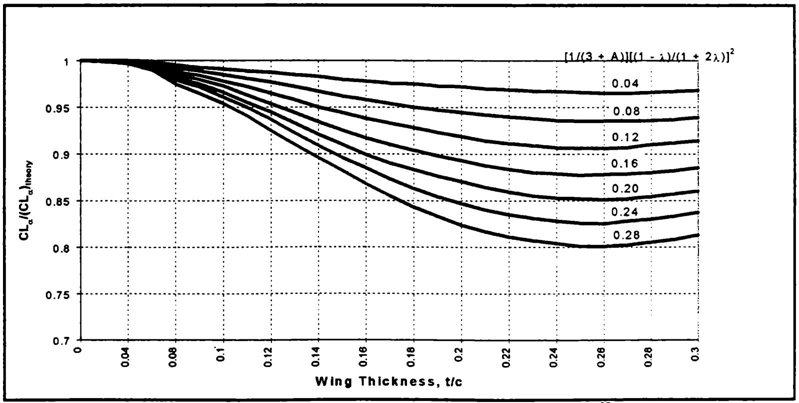


Figure 2: Wing Lift-Curve-Slope Thickness Correction Factor[10]

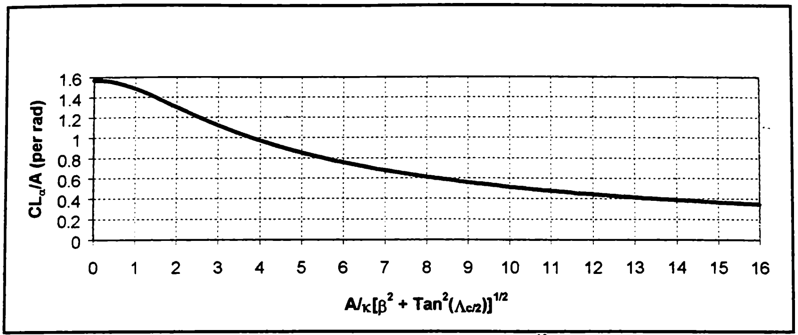


Figure 3: Subsonic Wing Lift-Curve Slope[10]

the Cartesian product quantization of the input-output state space through universally defined input and output fuzzy membership sets. Instead, each rule includes the definition of fuzzy input and output membership sets that are unique to the specific rule. Thus, the state space is partitioned into arbitrarily positioned fuzzy patches. Each fuzzy rule still defines relationships between the input state space and the output state space, but only for a region of the state space uniquely defined by that rule's membership sets. This approach to function modeling yields great flexibility and can improve the accuracy and size of the resulting function model.[9]

Function Modeling: Examples

Representative aerodynamic functions with one and two independent parameters respectively, were selected to investigate the use of fuzzy logic based function modeling. The Subsonic Wing Lift-Curve Slope[10] (Figure 3) was selected as a representative aerodynamic function with one independent parameter (1-D function). The Wing Lift-Curve-Slope Thickness Correction Factor[10] of Figure 3 was selected as a representative aerodynamic function with two independent parameters (2-D function).

Implementation and Timing

Membership Sets

The Centers method of defining a membership set requires the use of universally defined input sets. This requirement precludes their use with the Clustering inferencing method. The Gaussian method of defining membership sets uses the mathematically expensive exponential function. A comparison was made between the use of the exponential function to define this type of membership set and the use of a table look-up method to approximate the exponential function.

Inferencing Methods

For a 1-D function, the Sum-Product and Sum-Mean inferencing methods of Equations 7 and 8, reduce to equivalent implementations. Additionally, because the number of input fuzzy sets and consequents are the same, the FAM method and fuzzy clustering approach yield equivalent fuzzy rule definitions.

Comparison Results

Methodology: Comparisons were made between the effects of membership set definitions and method of inferencing for the different fuzzy systems described. A comparison was also made between the resources required for a linear table look-up and interpolation method and the fuzzy systems described for the one- and two-dimensional functions of Figures 2 and 3.

For these comparisons, the one-dimensional table was defined across the range from [0, 16] at an interval of 0.2. The two-dimensional table was defined across the first input variable range from [0, 0.30] at an interval of 0.01, and across the second input variable range at the curve values [0.04, 0.08, 0.12, 0.16, 0.20, 0.24, 0.28].

The function used to sample the table look-up and fuzzy models was intended to provide a balance between discontinuous steps across table breakpoints and the continuous sweeping across table breakpoints typical of most aerodynamic data. This function was defined for the one-dimensional models as:

```
for (Lim=1; Lim<=10*NumX; Lim++){
   for (i=0; i<=Lim; i++){
      XVal = 8 + 8*sin(i*2*PI/Lim);
   }
}
```

Where NumX is the number of breakpoints used to define the table look-up model. This function generated

321,200 data samples for the one-dimensional function comparisons.

The function to generate data samples for the two-dimensional models was similarly defined as:

```
for (XLim=1; XLim<=4*NumX; XLim++){
    for (YLim=1; YLim<=2*NumY; YLim++){
        for (i=0; i<=XLim; i++){
            XVal = 0.15 + 0.15*sin(i*2*PI/XLim);
            for (j=0; j<=YLim; j++){
                YVal = 0.16 + 0.12*sin(j*2*PI/YLim);
            }
        }
    }
}
```

Where NumX and NumY are the number of first and second input breakpoints used to define the table look-up model. This function generated 937,006 data samples for the two-dimensional function comparisons.

The fuzzy models for the one-dimensional function were generated with three, five and seven input membership sets. The two-dimensional models for the FAM and Clustering inferencing methods were also generated with three, five and seven input membership sets for the first input variable. The second input variable was modeled at the seven curve breakpoints defined. This resulted in a total of ten, twelve and fourteen input membership values to be evaluated with every data sample for the FAM model. The unique membership sets used by each rule of the Clustering model resulted in a total of twenty-one, thirty-five, and forty-nine membership values requiring evaluation with every data sample.

Membership Sets: The comparison of membership set definitions was determined using the Combs Union Matrix method for the one-dimensional and two-dimensional functions. The results of this comparison between the Centers, Center and Radius, and the Gaussian method using the Exponential function and an Exponential table are shown in Table 3. The Centers method of defining membership sets was used as the reference. The Centers method and the Center and Radius method are computationally equivalent. The advantage of the Center and Radius method being a more specific membership set definition and the ability to be used with all the inferencing methods presented. It is clear from this comparison, the Gaussian method incurs a significant computational performance penalty compared to the other two methods.

One Dimensional Function				
Set Number	**Centers**	**Center, Radius**	**Exp**	**Exp Table**
3	1.000	1.002	2.559	2.855
5	1.000	1.060	3.003	3.305
7	1.000	1.066	3.565	3.198
Two Dimensional Function				
Set Number	**Centers**	**Center, Radius**	**Exp**	**Exp Table**
3	1.000	0.938	3.317	3.671
5	1.000	1.040	3.941	4.411
7	1.000	1.083	4.083	4.594

Table 3: Membership Set Comparison

Inferencing Method: The comparison of inferencing method was determined using the Center and Radius set membership definition. The results of this comparison are presented in Table 4 for the two-dimensional function. The averaging operation of the Sum-Mean method incurs a ten percent penalty. The real stand-out of this comparison is the superior performance of the Combs Union Matrix which performed at least eight times faster than any of the other inferencing methods.

Two Dimensional Function				
Set Number	**Sum-Product**	**Sum-Mean**	**Time-Critical Optimized**	**Combs Union Matrix**
21/3	1.000	1.102	0.983	0.110
35/5	1.000	1.130	0.965	0.126
49/7	1.000	1.101	0.958	0.121

Table 4: Inferencing Method Comparison

Table Look-Up and Interpolation: The comparison between the fuzzy systems described and the linear table look-up and interpolation method were performed for both the one-dimensional and the two-dimensional function models. The results of this comparison are presented in Table 5. In general, the linear table look-up and interpolation models maintained a definitive

FAM, Cluster	**Combs Centers**	**Combs Centers, Radius**	**Set Number**
1.000	0.681	0.683	**3**
2.114	0.921	0.976	**5**
1.852	1.178	1.255	**7**

FAM	**Cluster**	**Combs Centers**	**Combs Centers, Radius**	**Set Number**
4.858	10.309	1.210	1.135	**3**
6.291	13.565	1.648	1.713	**5**
7.660	19.377	2.158	2.338	**7**

Table 5: Table Look-Up and Fuzzy Model Comparison

computational advantage over the fuzzy models examined in this study. This superiority is clearly demonstrated with the increased complexity of the two-dimensional function. Only the Combs Union Matrix method showed a potentially comparable performance with the table look-up method.

The numerous input membership values that were evaluated with every data sample for the two-dimensional FAM and Clustering models. This requirement to calculate numerous membership set values for every function evaluation appears to be the greatest handicap for real-time applications of fuzzy logic based function modeling.

Summary

This paper is part of a continuing investigation into the use of fuzzy logic based function modeling as an alternative to linear table look-up and interpolation in real-time flight simulation. Fuzzy systems have been shown to belong to the class of functions which are capable of approximating any real function to any desired degree of accuracy. These fuzzy systems are built upon very simple and efficient mathematical functions and can be implemented in a variety of forms. The choice of fuzzy set operators and resulting inferencing method can significantly affect the characteristics of a fuzzy system. This paper discussed implementation issues that need to be addressed when considering a fuzzy logic based function model for a real-time aerodynamic simulation. These issues consist of the method of representing the input and output fuzzy membership sets and the choice of inferencing method

Three methods of defining the set membership were considered: the Centers method, the Center and Radius method, and the Gaussian method. The Gaussian method incurs a significant computational performance penalty compared to the other two methods.

In a previous paper, three inferencing methods were presented as appropriate choices for accurate function modeling in a time critical environment: the Sum-Product, Sum-Mean and Time-Critical Optimized inferencing methods. Of these methods, the Sum-Mean inferencing method requires about ten percent greater computational resources. In this paper, a newly emerging inferencing method called the Combs Union Matrix method is introduced. This method proved to be 1.5 and 8 times faster for one- and two-dimensional functions respectively.

A timing comparison between these fuzzy based systems and a linear table look-up and interpolation algorithm was also presented. In timing comparisons with a linear table look-up and interpolation algorithm, only the Combs Union Matrix method showed a potentially comparable performance. The requirement to calculate numerous membership set values for every function evaluation, appears to be the greatest handicap for real-time applications of fuzzy logic based function modeling.

The future direction of these studies will examine methods to alleviate the computational burden of the set membership calculations. It is hoped this research will yield fuzzy systems that will be viable alternatives to the linear table look-up method in the future of real-time flight simulation.

Acknowledgements

I must express my sincere appreciation to my wife and children for the sacrifices they have made while I have pursued the research, tool development, and data collection necessary to make this paper possible. Without their support this would never have been accomplished.

References

1. G.J. Klir, T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, 1988.

2. H.-J. Zimmermann, *Fuzzy Set Theory And Its Applications*, Kluwer Publishers, 1991.

3. T. Takagi, M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Transactions on Systems, Man & Cybernetics*, Vol 15, pp. 116-132, 1985.

4. W.E. Combs, J.E. Andrews, "Combinatorial Rule Explosion Eliminated By A Fuzzy Rule Configuration," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 1-11, 1998.

5. D.D. Hensley, "Fuzzy Logic Based Function Modeling in Aerodynamic Simulations," Proceedings of the 1997 AIAA Modeling and Simulation Technologies Conference, pp. 296-304.

6. W.E. Combs, "Optimized Operators for Time-Critical Applications," Unpublished, 1994.

7. B. Kosko, "Fuzzy Systems As Universal Approximators," Proceedings of the 1992 IEEE International Conference on Fuzzy Systems, pp. 1153-1162.

8. B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, 1992.

9. H.R. Berenji, P.S. Khedkar, "Clustering in Product Space for Fuzzy Inference," Proceedings of the 1993 IEEE International Conference on Fuzzy Systems, pp. 1402-1407.

10. D.E. Hoak, D.E. Ellison, et al, *USAF Stability and Control DATCOM*, Flight Control Division, Air Force Flight Dynamics Laboratory, Wright Patterson Air Force Base, 1968 Edition.

AIAA-98-4163

IDENTIFICATION OF DORNIER-328 REVERSIBLE FLIGHT CONTROL SYSTEMS

Susanne Weiss[*], Wilhelm Gockel[*], Wulf Mönnich[†], Detlef Rohlf
Deutsches Zentrum für Luft- und Raumfahrt (DLR, German Aerospace Center)
Institute of Flight Mechanics, Postfach 3267, D-38022 Braunschweig, Germany

Abstract

A database for the high performance propeller aircraft Dornier-328 with reversible flight controls was identified from flight test data using system identification methods. The identified database, which includes both rigid-body aerodynamics and hinge moments, meets the accuracy requirements of a Level D flight simulator for this regional transport aircraft. This paper describes the procedures necessary to extract the aerodynamic hinge moments for the three primary controls from flight test data. The modeling process is described, that results in one degree-of-freedom systems for each control path with pilot force input and control surface deflection as output. The approach taken to arrive at a hinge moment database covering the entire operational envelope is outlined. Problems encountered in accounting for friction and in covering the low speed envelope where the aerodynamic forces are small are addressed. The applicability of the FAA-tolerances for the control dynamics validation tests to reversible flight controls and integrated testing with force input is questioned.

Nomenclature

Symbols

$\bar{c}$	mean aerodynamic chord
C_S	thrust coefficient
C_{SP}	slipstream coefficient
CH	hinge moment coefficient
F	thrust
f_{slip}	slipstream influence factor
Ma	Mach number
ME	sum of external moments
MF	friction moment
MH	aerodynamic hinge moment
$\bar{q}$	dynamic pressure
r	yaw rate
s	wing semispan
S	reference area
V	true airspeed
α	angle of attack
β	sideslip angle
δ_{RS}	roll spoiler deflection
$\Delta\delta_{RS}$	roll spoiler floating angle
$\Delta\xi$	aileron floating angle
η	elevator deflection
η_K	flap position
σ	ground effect influence factor
Θ	moment of inertia
ξ	aileron deflection
ζ	rudder deflection

Subscripts and superscripts

ail	aileron
elv	elevator
GE	ground effect
j	general index (left or right)
L, R	left, right
$prop$	propeller
rud	rudder
$slip$	slipstream
SE	single engine
ST	spring tab
TT	trim tab
unl	unloaded
0	zero moment

Introduction

The Dornier-328 (Fig. 1) is a high-wing, high-tail regional transport aircraft powered by two Pratt & Whitney Canada PW 119 turboprop engines[1]. The aircraft has an overall length of 69 ft 10 in,

[*] Member AIAA

[†] Affiliate Member AIAA

wingspan of 68 ft 10 in, and mean aerodynamic chord of 6 ft 8 in. The maximum performance is characterized by a takeoff and landing weight of 27558 lb, a rate of climb of 2420 ft/min, and a cruising speed of 335 KTAS. The aircraft has a maximum range of 700 nm with 30 passengers on board. In addition to the high speed capability at all altitudes, the Dornier-328 has short field takeoff and landing capabilities, which can be further improved by the use of optional ground spoilers.

All three primary flight controls are reversible, trimming is provided by trim tabs, and the rudder actuation is supported by a spring tab. The hydraulically operated roll spoilers are linked to the ailerons.

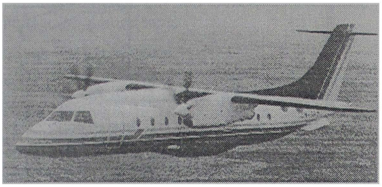
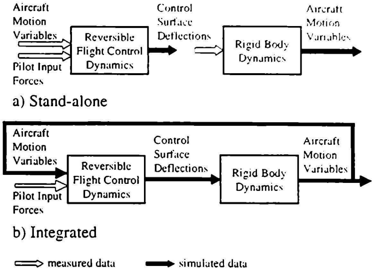


Fig. 1: Dornier-328

The Dornier-328 aircraft manufacturing program is supported by two training facilities providing six-axis motion based flight simulators. These flight simulators are to meet the Level D fidelity standards specified by the Federal Aviation Administration (FAA)[2], allowing pilot training and check-outs to be carried out entirely on the simulator.

The first flight training simulator for the Dornier-328 incorporating the wind-tunnel predicted database showed certain limitations in the simulation fidelity, especially for the reversible flight controls. The aircraft manufacturer Fairchild-Dornier, hence, subcontracted the DLR Institute of Flight Mechanics to generate a suitable database meeting the FAA Level D requirements.

Based on the experience at DLR[3], it was considered appropriate to generate a new database from flight data applying system identification rather than incrementally updating the wind-tunnel predictions. The output error method in the time domain was used to estimate the parameters of the nonlinear models for the rigid-body and the reversible control systems. The rigid-body aerodynamic database has been identified from a model driven by measured control surface deflections[4] (see Fig. 2a). In separate identifications, the hinge moment database for the primary controls (elevator, aileron, rudder) has been identified from pilot input force driven dynamic models of the flight controls. The development of these dynamic models including friction and the hinge moment identification from flight test data are covered in this paper.


Aircraft Motion Variables
Pilot Input Forces
Reversible Flight Control Dynamics
Control Surface Deflections
Rigid Body Dynamics
Aircraft Motion Variables
Aircraft Motion Variables
Pilot Input Forces
Reversible Flight Control Dynamics
Control Surface Deflections
Rigid Body Dynamics
Aircraft Motion Variables
measured data
simulated data

a) Stand-alone

b) Integrated

Fig. 2: Stand-alone and Integrated Models

Because of the reversible flight controls, it was mandatory to demonstrate the end-to-end match based on the integrated model with force inputs (see Fig. 2b). For this, the 6-DoF equations of aircraft motion are coupled with the dynamic models of the flight control systems driven through pilot applied forces (incorporating the identified rigid-body aerodynamic and hinge moment databases). The adequacy of the identified database was demonstrated by off-line simulation of the 170 flight maneuvers defined in the Acceptance Test Schedule (ATS) and an additional 1100 maneuvers at other flight conditions, thereby establishing the global validity.

At the time of writing this paper, the Dornier-328 flight training simulator utilizing the identified database has received certification from the German, French, and Dutch authorities. FAA certification is expected to follow soon.

Evaluated Flight Test Data

The flight test program for database generation was carried out by the aircraft manufacturer Fairchild-Dornier at Oberpfaffenhofen. Four different flap positions (0°, 12°, 20°, and 32°) and at least two speeds and three power settings (flight idle, power for level flight, and maximum takeoff power) for each flap position were investigated. At each of these trimmed flight conditions a sequence of dynamic maneuvers was carried out. It consisted of an elevator doublet exciting the short period motion, a large amplitude elevator pull and push, a bank-to-bank maneuver, an aileron step input, a rudder doublet exciting the dutch roll, and a rudder step input resulting in steady sideslip conditions. Other in-flight maneuvers were longitudinal trims, elevator, aileron, and rudder mistrims, climbs and descents, power change dynamics, gear change dynamics, control dynamics, longitudinal maneuver stability, static longitudinal stability, spiral stability, phugoid dynamics, and stalls.

Several fly-bys at different altitudes were carried out to enable identification of ground effect. Some tests were flown with forward and aft center of gravity locations affecting elevator trim, and left and right lateral cg-positions which affect the aileron angles required for trim. The dynamic maneuvers and the fly-bys were repeated with one engine inoperative for identification of single engine effects. Ground tests included accelerations and decelerations with and without reverse thrust, normal takeoffs, engine failure on takeoff as well as normal, single engine, and crosswind landings. All maneuvers were flown manually by the pilot. Data was recorded at a sampling rate of 50 Hz.

Dynamic Models

For the identification of the hinge moment database, each control system was modeled as a stand-alone system driven by pilot force and using the flight measured rigid-body variables as additional inputs (see Fig. 2a). For the validation of the integrated model, the same model structure was used only using the simulated aircraft states as calculated from the identified rigid-body model. In both cases, local variables such as airspeed and airflow angles, angular rates, and angular and linear accelerations are needed for each control system. They are calculated from the corresponding variables at the reference point by accounting for the positon vector of the respective control surface.

For the control systems, inertia effects due to aircraft motion are considered as external moments. These are caused by linear acceleration of the aircraft when the center of gravity of the control surface does not lie on its swivel axis (unbalanced mass effect) and by angular acceleration of the aircraft due to the moment of inertia of the control surface.

Elevator

In the elevator control system the pilot input forces are transmitted from the stick to the elevator by cables and rods (see Fig. 3). The system is preloaded by a downspring and, to prevent aircraft stall, a stick pusher acts if the angle of attack exceeds a limit that depends on flap setting.

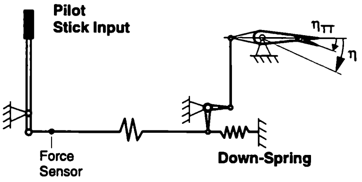


Fig. 3: Elevator Control System

For the dynamic model, the mass and moments of inertia of the connecting rods and cables are lumped together with those of the elevator surface (including trim tab) into an equivalent moment of inertia Θ_{elv}. As the pilot force is measured in the first rod, the stick inertia effects are already included in this input signal. This results in a quasi static transmission of pilot input force to the elevator axis. The elevator dynamics can therefore be formulated as a 1-DoF system using elevator deflection and deflection rate about the swivel axis as system states

$$\ddot{\eta} = \tfrac{1}{2}\left(ME_{elv} + MF_{elv}\right)/\Theta_{elv} \qquad (1)$$

The sum of external moments excluding friction ME_{elv} consists of the following components:

- Aerodynamic hinge moment (MH_{elv})
  This is a function mainly of elevator deflection, elevator trim tab deflection, velocity, and local flow angles. The identification of the hinge moment from flight test data is covered in the next section.
- Pilot forces transmitted to the aft part
  The input forces of the pilot generate pilot input moments by the stick lever arm in the front part

of the elevator control system. These moments are transmitted to the aft part by a nonlinear connection of cables and rods. Thus, the deflection ratio and hence the corresponding moment ratio varies with elevator deflection. It was provided by the aircraft manufacturer.

- Inertia effects due to aircraft motion
  These are calculated from the elevator static moments and moments of inertia using the local accelerations at the elevator swivel axis due to rigid-body motion.
- Downspring moment
  The downspring characteristic was provided by the manufacturer but had to be slightly modified based on ground tests.
- Stick Pusher Moment
  The stick pusher is modeled simply as a constant additional moment which is identified from stall maneuvers. Changes in stick pusher lever arm and spring force due to tension release as well as the spring reload phase are neglected.

The friction of the entire elevator control path is lumped into the friction moment MF_{elv} acting at the elevator swivel axis. Friction modeling aspects are covered in a separate chapter.

Aileron / Roll spoiler

The aileron control system consists of two connected paths (see Fig. 4), which are identical except for an aileron trim tab on the left side.

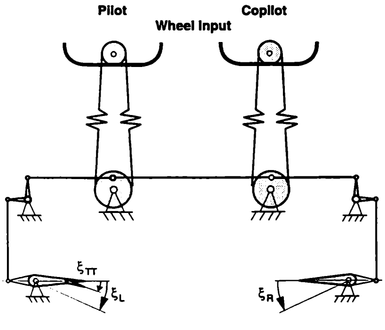


Fig. 4: Aileron Control System

The mass and moments of inertia of the connecting rods and cables, the wheel, and the aileron itself are again lumped into an equivalent moment of inertia Θ_{ail} acting at the aileron swivel axis. The dynamic system for the aileron can then be formulated as a 1-DoF system with right aileron deflection and deflection rate as system states

$$\ddot{\xi}_R = \tfrac{1}{2}\left(ME_{ail}^R + ME_{ail}^L + M_{defl} + MF_{ail}\right)/\Theta_{ail} \qquad (2)$$

For both sides, the sum of external moments excluding friction $ME_{ail}^j,\ j = L, R$ consists of the following components:

- Aerodynamic hinge moment (MH_{ail}^j)
  The hinge moment is mainly a function of aileron deflection, aileron trim tab deflection, velocity, and local flow angles and is identified from flight test data.
- Pilot forces transmitted to the aft part
  The input forces of left and right pilot generate pilot input moments by an effective wheel lever arm in the front part of the aileron control system. These moments are transmitted nonlinearly to the aft part by cables and rods. The moment ratio between front and aft part is therefore varying with aileron deflection. Values for the moment ratio were provided by the aircraft manufacturer.
- Inertia effects due to aircraft motion
  As for the elevator, these moments are calculated from the static moments and moments of inertia of the ailerons using the local accelerations and rates at the aileron swivel axes due to rigid-body motion.

During ground test evaluation, a small restoring moment was found to be present. It is modeled as an additional aileron deflection dependent moment

$$M_{defl} = C_{ail}(\xi_R - \xi_L)/2 \qquad (3)$$

with spring rate C_{ail}.

Similar to the elevator system, all system friction is lumped into a friction moment MF_{ail} acting at the aileron swivel axis.

Aileron floating

The deflection ratio of left to right aileron was provided by the manufacturer. This ratio, however, is only valid for the unloaded case. In flight, due to wing bending and the elasticity of the aileron control system in the wing, the aerodynamic hinge moments cause symmetric upward directed deflections on both sides.

The above described dynamic model of the aileron gives the resultant right aileron deflection including this floating. To arrive at the unloaded deflection, the floating effects have to be subtracted.

$$\xi_R^{unl} = \xi_R - \Delta\xi_R \tag{4}$$

The unloaded left aileron deflection is then derived from the unloaded right aileron using the above mentioned ratio. Elastic deflections due to aerodynamic loads of the left aileron itself and the trim tab must be added to arrive at the resultant deflection of the left aileron

$$\xi_L = \xi_L^{unl} + \Delta\xi_L \tag{5}$$

The floating effects $\Delta\xi_j$ are assumed to be directly proportional to dynamic pressure. They depend on the local airflow angles, Mach number, thrust setting, flap position, and for the left side additionally on the trim tab deflection. They are identified from flight test data together with the aerodynamic hinge moments.

Roll spoiler model

The deflections of the left and right roll spoiler depend on the respective aileron deflection. The spoilers are driven by a hydraulic actuator and are not mechanically linked to the ailerons. The ratio between aileron and roll spoiler deflection for the unloaded case has been identified from ground tests. Using the unloaded aileron deflections (as derived above), the unloaded roll spoiler deflection can thus be determined. Roll spoiler floating is taken into account by an increment $\Delta\delta_{RS.j}$ which is proportional to dynamic pressure and is a function of local flow angles, Mach number, and roll spoiler deflection itself.

$$\delta_{RS.j} = \delta_{RS.j}^{unl} + \Delta\delta_{RS.j} \tag{6}$$

Upon spoiler extension, the roll spoiler deflection rate is limited to 182 °/s. As no dynamic model for the roll spoilers is used for the identification, the deflection increment between two consecutive time points is limited instead of limiting the actual deflection rate.

Rudder and spring tab

The left and right pedals are connected to the so called pilot input lever through cables (see Fig. 5). The pilot input lever has a pivoted connection with the spring tab (capable of transmitting moments) and is linked to the rudder by a torsion spring. The spring tab is linked to the rudder trailing edge by a swivel. A rudder damper acts at the rudder axis.

The high frequency dynamics of the connecting rods and cables between front and aft part together with pilot input lever and spring tab are neglected and only the stationary state is considered. This results in a quasi static system for this part of the control system. The equations of motion are set up for the rudder with the rudder inertia Θ_{rud} accounting also for the spring tab. Based on these assumptions, the dynamics of the rudder control system can be reduced to a 1-DoF system with rudder deflection and deflection rate as system states.

$$\ddot{\zeta} = (ME_{rud} + MF_{rud}) / \Theta_{rud} \tag{7}$$

The sum of external moments ME_{rud} excluding friction consists of the following components:

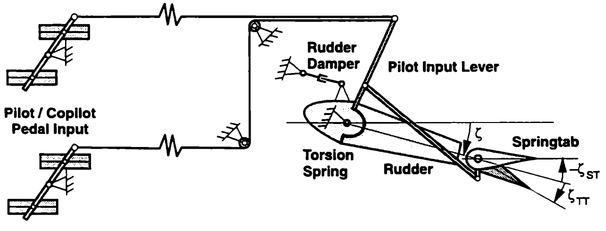


Fig. 5: Rudder Control System

- Aerodynamic hinge moment (MH_{rud})
  This is a function of rudder deflection, rudder trim tab deflection, velocity, local flow angles, and thrust.
- Spring tab moment
  The forces resulting from spring tab aerodynamic loads are transmitted to the rudder by the spring tab swivel. The identification of the rudder and spring tab hinge moments from flight test data is covered in the next section.
- Torsion moment
  This moment is due to torsion spring deflection and reflects the moment imparted to the rudder by the pilot input lever.
- Rudder damper moment
  The rudder damper moment is a function of rudder deflection rate and was provided by the manufacturer.
- Inertia effects due to aircraft motion
  These are calculated from the static moments and moments of inertia of the rudder using the local accelerations at the rudder swivel axis due to rigid-body motion.

All system friction is modeled by a friction moment MF_{rud} acting at the rudder swivel axis. Friction of the spring tab is neglected.

As the spring tab is not modeled as a separate dynamic system, the stationary spring tab deflection has to be calculated from the balance of moments at the pilot input lever, using given pilot input force, rudder deflection, and the ratio between spring tab and torsion spring deflection. The complex formulation of the spring tab hinge moment prevents an analytical solution, hence an iterative approach is applied.

Pilot input lever deflection is coupled with spring tab deflection and hence a result of this iteration. However, the deflection ratio between front and aft part was provided as a function of input lever deflection. Therefore, an additional outer iteration is required to determine the proper deflection ratio, and thus the required moment ratio, between pilot pedal input and pilot input lever

Hinge Moment Identification

The hinge moments for each control system are identified separately and independent of the identification of the rigid-body aerodynamic database. This leads to smaller models for parameter estimation, which is very important during the iterative phase of model building. Moreover, it also eliminates an adverse influence of any inaccuracies in the rigid-body aerodynamic database on the estimation of the hinge moments and vice versa.

Identification of the hinge moments has been carried out by postulating suitable derivative models in an analytical form. The complexity of the models was increased step by step, evaluating successively more flight test data at different test conditions. The identification yields four sets of derivatives for the different flap positions. As it was desired by the aircraft manufacturer to have the database in table form, the flight identified derivative database was afterwards converted into tables for implementation in the simulator.

Elevator

The elevator hinge moment is normalized with dynamic pressure, elevator area, and elevator reference chord

$$MH_{elv} = \tfrac{1}{2} CH_{elv} \bar{q} S_{elv} \bar{c}_{elv} \qquad (8)$$

Without listing all derivatives explicitly, the elevator hinge moment coefficient was identified as

$$\begin{aligned} CH_{elv} = {} & CH_{elv,0}(\alpha_{elv}, Ma, C_{SP}, \eta_K) \\ & + CH_{elv,\eta}(\alpha_{elv}, Ma, C_{SP}, \eta, \eta_K) \\ & + CH_{elv,\eta_{TT}}(\alpha_{elv}, Ma, \eta_{TT}, \eta_K) \\ & + CH_{elv,\dot{\alpha}}(\eta_K)\dot{\alpha}_{elv} \bar{c}_{elv} / V \\ & + CH_{elv,\dot{\eta}}(\eta_K)\dot{\eta} \bar{c}_{elv} / V \\ & + CH_{elv,\beta}(\beta_{elv}, \eta_K) \\ & + CH_{elv,GE}(\eta, \eta_K)\sigma_{elv} \end{aligned} \qquad (9)$$

where` the thrust influence is formulated using the slipstream coefficient

$$\begin{aligned} C_{SP,j} &= F_j / (2 S_{prop} \bar{q}_{slip,j}) \\ \bar{q}_{slip,j} &= \bar{q} + F_j / S_{prop} \end{aligned} \qquad (10)$$

The aerodynamic hinge moment of the elevator for the symmetrical flight configuration was identified from evaluating the dynamic maneuvers (elevator doublets and push-pull maneuvers), trims (normal and with forward and aft cg-position), and elevator mistrims. Considering the different speed and thrust levels, about 60 maneuvers were analyzed simultaneously for each flap position.

This led to a zero moment $CH_{elv,0}$ that is a function of local angle of attack, Mach number, and thrust setting. The elevator force gradient $CH_{elv,\eta}$ is modeled distinguishing between positive and negative deflections. Local angle of attack, Mach number, and thrust setting are the influencing parameters aside from elevator deflection itself. The elevator trim tab influence $CH_{elv,\eta_{TT}}$ on the hinge moment depends on local angle of attack, Mach number, and the trim tab deflection. An angle of attack rate influence $CH_{elv,\dot{\alpha}}$ was found and aerodynamic damping is provided by an elevator rate term $CH_{elv,\dot{\eta}}$.

Keeping these parameters fixed, additional effects were added on from separate identifications. Ground effect $CH_{elv,GE}$ was identified from fly-by maneuvers using the ground effect influence factor

$$\sigma_{elv} = 1 - \tanh(4 \cdot h_{elv} / 2 \cdot s) \qquad (11)$$

with h_{elv} being the height of the elevator above ground and s the wing semispan.

The influence of sideslip on the elevator hinge moment $CH_{elv,\beta}$ was identified from steady sideslip maneuvers and no additional modeling was required for the single engine cases.

Finally, it was necessary to fade out the elevator hinge moment for low speed ground operation. This was modeled as a linear function of true airspeed with the lower limit of the transition band set to 1 m/s and the upper limit identified from takeoff maneuvers to be 30 m/s.

Aileron / Roll spoiler

The aileron hinge moment is normalized with dynamic pressure, aileron area, and aileron reference chord

$$MH^j_{ail} = \tfrac{1}{2} CH^j_{ail} \bar{q} S_{ail} \bar{c}_{ail} \qquad (12)$$

The identified aileron hinge moment coefficients have the form

$$\begin{aligned} CH^j_{ail} = {} & CH^j_{ail.0}(\alpha^j_{ail}, C_{SP.j}, \eta_K) \\ & + CH^j_{ail.\xi}(Ma, C_{SP.j}, \xi_j, \eta_K) \\ & + CH^j_{ail.\xi_{TT}}(\alpha^j_{ail}, Ma, C_{SP.j}, \xi_{TT}, \eta_K) \\ & + CH^j_{ail.\beta}(\beta, \eta_K) \\ & + CH^j_{ail.\delta_{RS}}(\delta^j_{RS}, \eta_K) \\ & + CH^j_{ail.\dot{\alpha}}(\eta_k) \dot{\alpha}^j_{ail} \bar{c}_{ail} / V \\ & + CH^j_{ail.\dot{\xi}}(\eta_k) \dot{\xi}_j \bar{c}_{ail} / V \end{aligned} \qquad (13)$$

where the deflection rate of the left aileron is set to the negative value of the right aileron

$$\dot{\xi}_L = -\dot{\xi}_R \qquad (14)$$

This amounts to neglecting nonlinearities in the aileron kinematics as well as the elasticity of the aileron control system part in the wing.

The aileron hinge moment as well as the floating effects for aileron and rollspoiler were identified from dynamic maneuvers (aileron step, dutch roll, bank-to-bank) both in symmetric and single engine conditions, steady sideslip maneuvers, aileron mistrims, and trims with left and right cg. Thrust influences are modeled with the left and right slipstream coefficient (see eq. (10)). By assuming that the left engine influences only the left aileron and vice versa, the single engine cases could be dealt with together with the symmetric cases.

The identified zero moment $CH^j_{ail.0}$ is a function of local angle of attack and angle of sideslip as well as the respective thrust setting. The aileron force gradient $CH^j_{ail.\xi}$ is modeled distinguishing between positive and negative deflections. Mach number and thrust setting are the influencing parameters aside from aileron deflection itself. The aileron trim tab influence $CH^j_{ail.\xi_{TT}}$ on the aileron hinge moment depends on local angle of attack, Mach number, thrust setting, and the trim tab deflection. The respective roll spoiler deflection also has an influence $CH^j_{ail.\delta_{RS}}$. Sideslip and angle of attack rate influences are covered by $CH^j_{ail.\beta}$ and $CH^j_{ail.\dot{\alpha}}$. Aerodynamic damping is provided by an aileron rate derivative $CH^j_{ail.\dot{\xi}}$.

Rudder / Spring tab

The hinge moments of rudder and spring tab are normalized with dynamic pressure and rudder respectively spring tab area and reference chord. Unlike for elevator and aileron, slipstream influences proved to be very significant at low airspeed and are modeled by an additional coefficient (CH_{rud}^{slip} resp. CH_{ST}^{slip}) that is multiplied by the slipstream influence factor f_{slip}. This factor is 1 for low dynamic pressure and decreases to 0 with increasing dynamic pressure.

$$MH_{rud} = (CH_{rud} + CH_{rud}^{slip} \cdot f_{slip})\bar{q}S_{rud}\bar{c}_{rud} \qquad (15)$$

$$MH_{ST} = (CH_{ST} + CH_{ST}^{slip} \cdot f_{slip})\bar{q}S_{ST}\bar{c}_{ST} \qquad (16)$$

The basic rudder and spring tab hinge moment coefficients for symmetric flight configurations were identified from dynamic maneuvers (dutch rolls, aileron steps, steady sideslips) and rudder mistrims. As the slipstream effects at low airspeed are modeled separately, thrust is taken into account using the thrust coefficient

$$C_{S.j} = F_j / (S\bar{q}) \qquad (17)$$

Keeping the parameters identified from the symmetric flight conditions fixed, single engine effects were then identified from rudder and aileron input maneuvers with one engine inoperative as well as from single engine trims. The single engine effects are modeled as a function of the differential thrust coefficient

$$\Delta C_S = C_{S.R} - C_{S.L} \qquad (18)$$

They turned out to be highly asymmetric.

With r_{rud} being the yaw rate around the rudder swivel axis, the breakdown of the identified hinge moment coefficients is as follows

$$\begin{aligned} CH_{rud} = {} & CH_{rud.0}(\beta_{rud}, Ma, C_S, \eta_K) \\ & + CH_{rud.\zeta}(\beta_{rud}, Ma, \zeta, C_S, \eta_K) \\ & + CH_{rud.\zeta_{TT}}(\alpha_{rud}, \zeta_{TT}, \eta_K) \\ & + CH_{rud.\dot{\zeta}}(Ma, \eta_K)\dot{\zeta}\ s / V \\ & + CH_{rud.r}(\eta_K) r_{rud}\ s / V \\ & + CH_{rud.SE}(\beta_{rud}, \zeta, \zeta_{TT}, \Delta C_S, \eta_K) \end{aligned} \qquad (19)$$

$$\begin{aligned} CH_{ST} = {} & CH_{ST.0}(\beta_{rud}, Ma, \alpha_{rud}, C_S, \eta_K) \\ & + CH_{ST.\zeta_{ST}}(\beta_{rud}, Ma, \zeta_{ST}, \eta_K) \\ & + CH_{ST.\dot{\zeta}}(\eta_K)(r_{rud} + \dot{\zeta})\ s / V \\ & + CH_{ST.\zeta}(Ma, \zeta, \zeta_{ST}, \zeta_{TT}, \eta_K) \\ & + CH_{ST.SE}(\beta_{rud}, \zeta, \zeta_{ST}, \Delta C_S, \eta_K) \end{aligned} \qquad (20)$$

Slipstream influences were identified from ground tests, namely takeoffs and landings as well as accelerations and decelerations, both for symmetric thrust and for single engine cases. As the measured airflow angles for these low speed cases were very inaccurate, the integrated model (rigid-body plus control systems) had to be used for the identification of the slipstream effects. The identified hinge moment coefficients due to slipstream can be summarized as

$$\begin{aligned} CH_{rud}^{slip} = {} & CH_{rud.0}^{slip}(\beta_{rud}, \zeta, C_S, F, \eta_K) \\ & + CH_{rud.SE}^{slip}(\beta_{rud}, \zeta, \Delta C_S, F, \eta_K) \end{aligned} \qquad (21)$$

$$CH_{ST}^{slip} = CH_{ST.0}^{slip}(\zeta_{ST}, F, \eta_K) \qquad (22)$$

The identification of hinge moments for low dynamic pressure with a force input model is very problematic, as already small hinge moment errors may produce large deflection errors. Furthermore, the calculated local airflow angles (angle of attack and sideslip) provide only insufficient information about the actual flow at the rudder surface which is, in particular at low speeds, largely influenced by propeller slipstream.

Friction

Due to the fact that pilot forces are the input for the control system models, friction has to be taken into account. As no further information was available about the friction in the system, it was modeled as an additional moment acting at the corresponding control surface axis.

Friction was modeled with a very simple model including static and sliding (Coulomb) friction (see Fig. 6). With the respective control surface in standstill, static friction causes stiction until the external moments exceed the static friction moment (breakout). Once the control surface is in motion, there is a transition between static and sliding friction, which depends on deflection rate. Viscous friction was not included.

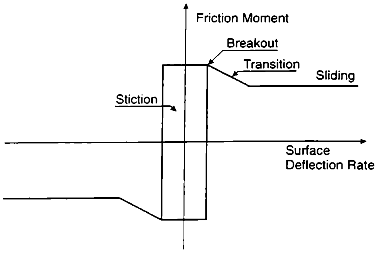


Fig. 6: Friction Model for Identification

Modeling friction with this simple model caused several problems, not all of which could be solved satisfactorily within the course of the Dornier-328 database development.

First of all, it was attempted to match all tests with the same friction model. This was very difficult because the static control tests on ground with engines off showed a different behavior than the tests in-flight. So, at least for the rudder system, two sets of friction moments had to be identified. The transition between the two levels of friction was modeled as a function of dynamic pressure, but it is probably a function of vibration level and should be modeled accordingly.

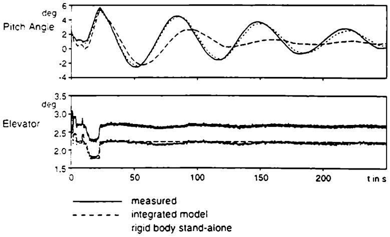


Fig. 7: Phugoid Motion

The stiction was necessary to model the ground tests with engine off, but caused problems in matching small amplitude motions in flight. Fig. 7 shows the phugoid validation test as an example. Due to static friction, the simulated elevator is at standstill after the initial input, whereas the real elevator is floating with a very small amplitude. The elevator match itself is very good, but the unmodeled floating causes large deviations in pitch angle for the integrated model. Additionally shown is the match for the rigid body stand-alone model that is driven by the measured elevator deflection plus a trim offset. Comparison of the three cases shows that frequency and damping of the phugöid are very sensitive to these small elevator movements.

Research in robotics indicates that the effective static friction is zero in systems with vibration[5]. So, the stiction area should probably be abolished for the cases with running engine. Furthermore, for all three controls, the identified static friction was lower than the identified sliding friction, which also indicates the inadequacy of the model.

Another problem caused by friction was encountered during the identification applying the gradient based Modified Newton-Raphson optimization method[3]. Due to the strong nonlinearities in the friction, the numerically determined gradients can be inaccurate, leading to abrupt termination of the optimization due to a singular information matrix, to convergence problems, and/or to local minima solutions. Time consuming optimization by hand was necessary in many of these cases. A possible solution would be to use an optimization method that does not use gradients, e.g. the simplex method as suggested in ref. 5.

It has to be kept in mind, however, that these friction models were used for the identification and validation of the hinge moment database only. In the final simulator, there is some friction caused by the hardware. Therefore, the identified friction can only be used as an initial guess and the simulator manufacturer uses his own friction model taking into account the friction of the simulator hardware.

Control Dynamics

In order for a simulator to be representative, it must present the pilot with the feel of the respective airplane. Compliance with this requirement is assessed by comparing the control feel dynamics of simulator and airplane for the different configurations. FAA required tests to verify the correct control dynamics are free responses to a step input (validation tests 2B1, 2B2, and 2B3 in ref. 2).

As these tests are conducted with hands off, no information about pilot feel can really be gathered. The control dynamics maneuver is useful for an aircraft with *irreversible* controls where a unique relation between pilot input force and position exists. For these aircraft, all other acceptance tests for the simulator are validated using aerodynamic control surface position as input. Validating the control dynamics with force input can then be used to verify correct force feel for the pilot. For an aircraft with *reversible* control systems, all tests of the Acceptance Test Schedule are validated using pilot force input, therefore already ensuring correct feel throughout the flight envelope.

Furthermore, the FAA tolerances for these tests are very stringent. For an underdamped response, there is a 10% tolerance on the time to the first zero crossing, a 20% tolerance on the frequency of the first oscillation period, and a 10% tolerance on the amplitude of the first overshoot (20% for the subsequent overshoots). The simulator must show the same number of significant overshoots to within one when compared to the aircraft. For a critically damped or overdamped response, there is a 10% tolerance on the time to reach 90% of the steady state value[2]. As it depends on the magnitude of the overshoots, the 10% tolerance on the amplitude of the first overshoot required the stick position errors to be less than 0.3° for the free response of the Dornier-328 elevator control system, which is in the order of the measurement accuracy.

These tolerances are very hard to meet for an airplane with reversible controls. For such a system not all of the response is deterministic and can be modeled. Part of the response is caused by friction which is influenced by vibration, temperature, loads, wear, etc. and cannot be modeled in full detail. As friction changes between various aircraft as well as during the lifetime of one aircraft, the simulator can never be representative for all aircraft of the same type.

To illustrate the strongly nonlinear behavior caused by friction, a 2nd order system with input u and output x was simulated

$$\ddot{x} = M_x x + M_{\dot{x}} \dot{x} + M_u u + MF$$
$$M_x = -10, \quad M_{\dot{x}} = -2, \quad M_u = 10$$

where the friction moment MF was modeled as a static friction of 1 and a coulomb friction of 0.128 (see Fig. 6). In Fig. 8 the response of the system to a unit step release input is shown. The shape of the response with only one or two overshoots is similar to those found for the elevator control dynamics of the Dornier-328.

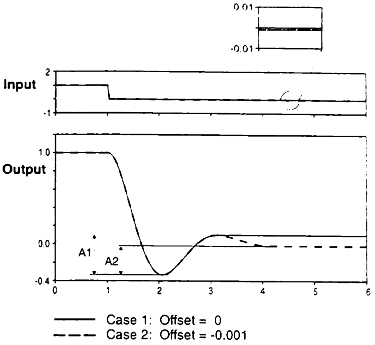


Fig. 8: 2nd Order System with Friction

The two cases differ only by a very small constant offset of 0.001 in the input, but the system response is very different. Due to the friction, the small input offset leads to large offsets in the stationary values of the response. As the amplitudes of the overshoots (A1 and A2 in Fig. 8) are determined with respect to the stationary response, they are different for the two cases. If this example were a control dynamics validation test, the second response would not be within tolerance of the first (and vice versa). These aprupt changes in system behavior make the determination of force offsets to match a validation test very difficult.

The same difference in system response can also be reached by lowering the coulomb friction from 0.128 to 0.127. Therefore, a slight variation in control system friction from one validation test to the other can make it impossible to fulfil the FAA tolerances for both cases with the same friction model.

Based on the experience gathered in modeling the Dornier-328 control systems, the authors think that the 2B-tests are not suitable for aircraft with reversible controls and are not necessary, at least as long as all other tests are validated with pilot force inputs. It is therefore suggested that the inclusion of these tests for reversible flight controls be critically reviewed or at least that the tolerances be relaxed significantly to realistic limits.

Concluding Remarks

This paper provides an overview over the development of the dynamic models for the reversible flight controls of the Dornier-328 including friction and hinge moments. The model development as well as the approach adopted to identify the hinge moment database from flight test data are elaborated. Problems that were encountered during the course of the project are discussed. Global validity of the identified database has been demonstrated on a very large number of flight maneuvers covering the entire operational envelope.

The difficulties arising from the demand to match some of the FAA requirements are discussed with respect to their adequacy for aircraft with reversible flight controls.

Acknowledgments

The authors would like to thank Mr. A. Straile and Mr. T. Otte of Dornier GmbH for discussions on the flight control systems and the measured data. The authors are also thankful to the aircraft manufacturer Fairchild-Dornier for having granted the permission to publish the investigations reported in this paper.

References

[1] Anon.: *Dornier-328 (Product Information).* Dornier Luftfahrt GmbH, Germany, March 1992.

[2] Anon.: *Airplane Simulator Qualification.* FAA Advisory Circular, AC 120-40B, Interim Version, Feb. 1991.

[3] Hamel, P.G., Jategaonkar, R.V.: *Evolution of Flight Vehicle System Identification.* Journal of Aircraft, Vol. 33, No. 1, Jan.-Feb. 1996, pp. 9-28.

[4] Jategaonkar, R.V., Mönnich, W.: *Identification of Do-328 Aerodynamic Database for a Level D Flight Simulator.* AIAA Modeling and Simulation Technologies Conference, Paper No. 97-3729-CP, New Orleans, LA, Aug. 11-13 1997.

[5] Cheok, K.C., Hu, H. Loh, N.K.: *Modeling and Identification of a Class of Servomechanism Systems with Stick-Slip Friction.* ASME Transactions, Journal of Dynamic Systems, Measurement and Control, Vol. 110, pp. 324-328, 1988.

[6] Dupont, P.E.: *The Effect of Friction on the Forward Dynamics Problem.* Int. Journal of Robotics Research, Vol. 12, No. 2, Apr. 1993, pp. 164-179.

AIAA-98-4164

THE NOVEL METHOD FOR IDENTIFICATION OF AIRCRAFT TRAJECTORIES IN 3-D SPACE

Artur KORGUL*

Military University of Technology, Warsaw, Poland
01-489 Warszawa, ul. Kaliskiego 2

ABSTRACT

In this paper the novel method for identification of aircraft trajectories in 3-D space is considered. The differential equations of the vector field of an aircraft's velocity are presented and analyzed. The inverted approximation method is proposed for estimating the smooth trajectories from the discrete and sparse past position readings of an aircraft in order to identify the state variables and kinematic parameters of an aircraft's movement. The linear and vectorizable procedure as well as its TDNN (time delay neural network) implementation for purpose of the trajectory recognition is outlined. The performance of this method is assessed in the simulation of maneuvering target interception.

INTRODUCTION

The main purpose of the tracking system for air defence is the estimation of target trajectories in the controlled area and their prediction into the near future.

Mathematical modeling of the target tracking process has always been a subject of extensive studies.[1,2] The aircraft's trajectory is usually recognized by means of statistical approach to time series analysis of radar plots.

Since the early sixties the Kalman filter and its variants have preferably been used for tracking applications.[1,2,3] In this approach the state of the tracked aircraft consists of its position and the time-derivatives of displacement. This displacement of an arbitrarily maneuvering aircraft would, in general, have a number of non-zero derivatives. An accurate model of the aircraft motion should include all these derivatives. The aircraft motion models of the highest order currently available in tracking literature include terms up to the second derivatives of the aircraft position.[2,3] These tracking techniques are well suited for the recognition of the straight line or moderately curved trajectories, whereas modeling of the modern generation of highly maneuvering aircraft call for better tracking performance than what is provided by acceleration models. The alpha-beta and alpha-beta-gamma trackers or even bank filters with the switching of structure represent classical approaches to solving this problem.[4,5] The reason for the inadequate tracking performance of current models is that higher order derivatives in the case of very highly maneuvering aircrafts are not insignificant, leading to model inaccuracies when terms only up to the aircraft acceleration are included.[3] As a result they do not easily comply with the physics of a real aircraft flight in aerodynamically dissipative environment. In order to formulate the essential assumptions for tracking the tactical aircraft trajectory, a deep analysis of aircraft flights is needed from the point of view of ground observer (i.e. radar system). Considering the real aircraft's flight in aerodynamically dissipative environment it is important to realize that it should be seen as a complex interaction between two main processes:

- the movement of an aircraft treated as a rigid body with longitudinal plane of symmetry and having its own propulsion (occurred to fuel consumption by the jet's engine and following the aircraft's mass reduction during flight in constant gravitational field), and
- the dynamic interaction between the air flow and the aircraft's profile resulting in aerodynamic drag and lift forces resisting and balancing, the aircraft's movement and intensification of fuel consumption.[6,7]

Simultaneously, the aircraft's movement along an arbitrary trajectory is for the radar tracking system (from the measuring point of view) nothing other but an „optical flow" of a visible particle and its vector field of velocity in 3-D space.[8,9] Consequently, the tracking procedures must appreciate additional constraints including: the total energy conservation during an aircraft's flight, the rectafiability of trajectory and coordinated turns of aircraft, if the vector relations between the state variables and kinematic parameters are to be properly identified for the feasible process of flights. For this purpose each unknown trajectory of an aircraft (i.e. a rectifiable continuous and smooth curve in 3-D space) should be considered as a result of a stable process of aircraft flight in aerodynamically dissipative environment (i.e. it must be seen as the coupled Hamiltonian system of the moving flat and symmetrical rigid body with its own propulsion and adjacent air flow generated around the aircraft's profile).[8]

Apparently, any fast disturbances are not essential for the perfect study of this flow phenomenon, as a relatively long period of past observations (of at least

[*] Senior Expert, Institute of Command System Automation

about ten past chaotic samples of position readings) is needed. The smooth solutions are rather interesting in the tracking practice, so the noiseless model would be even more appropriate to describe the ideal aircraft's trajectory by means of low-order differential equations in 3-D space.[7, 9] In recent years, considerable progress has been made in modeling chaotic time series with neural networks. Though neural networks are not a panaceum, they can provide significant advantages over classic techniques and can also be implemented for real-time solutions.[10, 11] In this paper the mathematical modeling and simulation has been used to develop a new method of recognizing highly maneuverable aircraft trajectories in 3-D space. The differential equations of the vector field of an aircraft's velocity are presented and analyzed. The inverse approximation (according to Polish AK method) has been adapted for the estimation of the smooth trajectories in 3-D space.[12, 13, 14] The vectorizable procedure has been developed for the identification of the state variables and kinematic parameters of aircraft movement. The recurrent network of TDNN (time delay neural network) type has been proposed for the estimation of the trajectory in 3-D space.[15] Several simulation experiments have been carried out to prove the usefulness of this methodology. For this purpose the pursuit - evasion scenario has been arranged and the time series of numerical data has been artificially generated to model the successive measurements of aircraft position in 3-D space geometry in order to supply on-line the tracking procedures fundamental in solving the task of target interception. Furthermore, modeling errors have been estimated to justify the sufficient degree of simulation fidelity. The simulation results show that effective tracking of a highly maneuverable aircraft requires implementations of differential filters of TDNN type (preferably designed at VLSI - very large scale of integration technology). The paper covers the following topics:

- mathematical modeling of aircraft trajectory in 3-D space geometry;
- discussion of the proposed model;
- statement of the identification problem;
- solution to the identification problem;
- error estimation;
- application of TDNN to trajectory recognition;
- simulation experiments;
- conclusions.

MATHEMATICAL MODELING OF AIRCRAFT TRAJECTORY IN 3-D SPACE

Apparently, the rigid body movement (i.e. aircraft) is well describable by Hamiltonian equations, but for practical purposes of aircraft trajectory tracking it is too complicated a mathematical description.[8]

On the other hand, according to energy conservation, the total fuel consumption during an aircraft's flight along an arbitrary trajectory is determined by the balansing of the energy dissipation (mainly to aerodynamic drag) and mechanical energy accumulation (due to the variation of velocity and altitude of an aircraft's flight). In other words, energy (supplied with the fuel to the system) is wholly distributed onto the generation and sustaining of the observed (by the radar) fictive „optical flow" of vector velocity field of the aircraft.[9] As a result it may be well modeled as the Bernoulli-like fictive flow of a thin stream of an ideal liquid in 3-D space along this trajectory.[8] Consequently, the succeeding past positions of the aircraft along its trajectory can be interpreted as a continuum of „visible particles" of a fictive „optical flow" within the observation area of the radar system.

Thereupon, the aircraft's trajectory, which can be recognized by the ground radar tracking system, should consist of an arbitrary continuous and smooth curve additionally constrained by the following conditions:

- in kinematic sense, the aircraft's velocity vector, the aircraft's acceleration vector and the aircraft's angular rate vector at each point of trajectory should together define the Lie algebra in 3-D Euclidian space, and
- in geometric sense, the trajectory should be a geodesic line in this space (i.e. be rectifiable).[8]

To consider all these aspects (discussed above) the appropriate frame of reference must be defined and the aircraft (a flat rigid body with its own propulsion) should be modeled as a single weighted particle with a velocity vector, attached to the weightless lifting plane (in its center of gravity). Next, the movement of thus defined airplane is examined in an aerodynamic environment during a finite period of observation $[t_0, t_0 + T_p]$ in a right handed inertial frame of reference $Oxyz$ (i.e. Cartesian coordinate system). Furthermore, the two additional frames of reference $(Cx'y'z')$ and $(Cx''y''z'')$, which are attached to the moving particle (i.e. center of gravity of airplane), are considered, and are defined as follows (see Fig. 1):

- for the $(Cx'y'z')$ frame of reference, the x', y' and z' axes are parallelly oriented to the x, y and z axes of the inertial frame of reference $(Oxyz)$;
- for the $(Cx''y''z'')$ frame of reference, the x'', y'' and z'' axes have Frenet frame configuration, such that the x'' axis is collinear with the vector of linear velocity of maneuvering aircraft and it always lies on the strictly tangential surface (of the Frenet frame) to the trajectory of aircraft's movement, the y'' axis is co-linear with the vector of centripetal force acting on maneuvering aircraft and the z'' axis is co-linear with the vector of the angular rate of a maneuvering aircraft.

Note: In thus defined frames of reference, the Euler angles (ψ,θ,φ) (i.e. yaw angle, flight-path angle and bank angle) naturally describe the orientation of the

Frenet frame (and the „lifting plane" of aircraft $Cx''y''$) with respect to the inertial frame of reference ($Oxyz$).

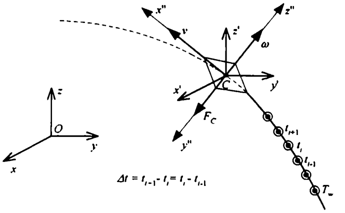


Fig. 1. The configuration of reference frames

The arbitrary trajectory of a maneuvering aircraft can be now described in ($Oxyz$) frame of reference by the following set of equations:

$$\begin{cases} x(t+\Delta t) = x(t) + v_x(t)\cdot\Delta t \\ y(t+\Delta t) = y(t) + v_y(t)\cdot\Delta t \\ z(t+\Delta t) = z(t) + v_z(t)\cdot\Delta t \\ \psi(t+\Delta t) = \psi(t) + \omega_z(t)\cdot\Delta t \end{cases} \tag{1}$$

$$\text{for} \quad t \in \left[t_0, t_0 + T_p\right]$$

where:

T_p - the possible horizon of trajectory prediction,

Δt - the time step of update.

Whereas the kinematic relations (consisting of the nonlinear state equations of an aircraft's model of flight in ($Oxyz$) frame of reference) take the form:

$$\begin{cases} \dfrac{dx}{dt} = v_x(t) = v(t)\cdot\cos[\theta(t)]\cdot\sin[\psi(t)] \\ \dfrac{dy}{dt} = v_y(t) = v(t)\cdot\cos[\theta(t)]\cdot\cos[\psi(t)] \\ \dfrac{dz}{dt} = v_z(t) = v(t)\cdot\sin[\theta(t)] \\ \dfrac{d\psi}{dt} = \omega_z(t) = -\dfrac{g\cdot \mathrm{tg}[\varphi(t)]}{v(t)\cdot\cos[\theta(t)]} \end{cases} \tag{2}$$

g - acceleration due to gravity;

and the geometric relations (defining the natural nonlinear output equations of an aircraft's model of flight in ($Oxyz$) frame of reference) take the form:

$$\begin{cases} v(t) = \sqrt{v_x^2(t) + v_y^2(t) + v_z^2(t)} \\ \theta(t) = \mathrm{arctg}\dfrac{v_z(t)}{\sqrt{v_x^2(t) + v_y^2(t)}} \\ \psi(t) = \mathrm{arctg}\dfrac{v_x(t)}{v_y(t)} \\ \varphi(t) = \mathrm{arctg}\dfrac{v(t)\cdot\omega_z(t)}{\dot{v}(t)} \end{cases} \tag{3}$$

$$\text{for: } \dot{v}(t) \neq 0;\ v_x(t) \neq 0;\ v_y(t) \neq 0;$$

and

$$\dot{v}(t) = \sqrt{\dot{v}_x^2(t) + \dot{v}_y^2(t) + \dot{v}_z^2(t)} \tag{3a}$$

The trajectory of aircraft flight is fully defined by the set of equations (1), (2) and (3) and these relations seem to be similar to the model of a differentially flat system which can be well used for aircraft trajectory prediction and generation.[6] Strictly speaking, they could be considered as the differentially flat system, if it was possible to express the angular rate $\omega_z(t)$ by means of velocity components $\{v_x(t), v_y(t)\}$ and their higher order time derivatives.

The answer to the question, „how to do it", is revealed further in the paper.

DISCUSSION ON THE MATHEMATICAL MODEL

According to Euler theory, the group of revolutions in 3-D Euclidean space (provided with the left-invariant Riemannian metric) is sufficient to express an arbitrary movement of a rigid body along a geodesic line.[8]

The Euler theory is true not only for rigid body movement, but it may also be considered in hydrodynamics of ideal liquid flows. For this purpose, a group of transformations should consist of a group of volume conserving diffeomorphisms in flow area. In this case, the least action principle means that the movement of ideal liquid takes place along the geodesic line of right-invariantly defined metric (i.e. kinetic energy).[8] Accordingly, the Euler equations of ideal liquid flow in hydrodynamics are analogous to Euler equations of rigid body movement, and especially for aircraft movement in aerodynamically dissipative environment, where the energy conservation law insists on the equalization between right -invariant metric (for airflow around an aircraft) and left-invariant metric (for aircraft movement). As a result, the trajectory of aircraft movement in the finite area of 3-D space may be presented (for tracking purposes) by means of Bernoulli - like partial differential equations of fictive flow of ideal liquid (e.g. „optical flow" of velocity vector field of aircraft).[8, 9] Practically, in this case, the area of trajectory observation may be organized as a moving window (i.e. a three dimensional shift register with the time width of T_w) being supplied serially (with the time spacing of Δt) with the data of succeeding 3-D position readings of the aircraft along its trajectory (including the actual position and the number of past positions - see Fig.1).

Thereupon, the set of kinematic equations (2) and (3) can be compared (at the front of flow area for the actual time instant) with the equations of the flow of the velocity vector field considered in the finite period of past trajectory observations as follows:

$$\begin{cases} \dfrac{dx}{dt} = v_x(x,y,z,\tau)_{|\tau=0} \\ \dfrac{dy}{dt} = v_y(x,y,z,\tau)_{|\tau=0} \\ \dfrac{dz}{dt} = v_z(x,y,z,\tau)_{|\tau=0} \\ \dot{v}_x(t) = \dfrac{\partial v_x}{\partial \tau}_{|\tau=0} - 2\omega_z(x,y,z,\tau)_{|\tau=0} \cdot v_y(x,y,z,\tau)_{|\tau=0} + \\ \qquad + 2\omega_y(x,y,z,\tau)_{|\tau=0} \cdot v_z(x,y,z,\tau)_{|\tau=0} \\ \dot{v}_y(t) = \dfrac{\partial v_y}{\partial \tau}_{|\tau=0} + 2\omega_z(x,y,z,\tau)_{|\tau=0} \cdot v_x(x,y,z,\tau)_{|\tau=0} - \\ \qquad - 2\omega_x(x,y,z,\tau)_{|\tau=0} \cdot v_z(x,y,z,\tau)_{|\tau=0} \\ \dot{v}_z(t) = \dfrac{\partial v_z}{\partial \tau}_{|\tau=0} + g - 2\omega_y(x,y,z,\tau)_{|\tau=0} \cdot v_x(x,y,z,\tau)_{|\tau=0} + \\ \qquad + 2\omega_x(x,y,z,\tau)_{|\tau=0} \cdot v_y(x,y,z,\tau)_{|\tau=0} \end{cases} \tag{4}$$

for: $x(t,\tau) = x(t-\tau)$;

$y(t,\tau) = y(t-\tau)$;

$z(t,\tau) = z(t-\tau)$; $\quad \tau \in [0, T_w]$;

and

$$2\vec{\omega} = \text{rot}\,\vec{v} \tag{5}$$

where: T_w means the width of the time moving window for trajectory observations.

Equations (4) simply describe both the rotational and translational flow of the vector field of the aircraft's velocity for the past, whereas the kinematic equations (1), (2) and (3) show the same for the present and near future time instant.

Nevertheless, the main advantage of equations (4) and (5) lies in the fact that they suggest how to identify the angular rate $\vec{\omega}(t)$ of aircraft during its movement along the curvilinear trajectory. For this purpose we assume that the angular rate $\vec{\omega}(t)$ should be expressed by means of the following equations:

$$\begin{cases} \omega_x(t) = \dfrac{1}{2}\left[\dfrac{\dot{v}_z(t)}{v_y(t)} - \dfrac{\dot{v}_y(t)}{v_z(t)}\right] \\ \omega_y(t) = \dfrac{1}{2}\left[\dfrac{\dot{v}_x(t)}{v_z(t)} - \dfrac{\dot{v}_z(t)}{v_x(t)}\right] \\ \omega_z(t) = \dfrac{1}{2}\left[\dfrac{\dot{v}_y(t)}{v_x(t)} - \dfrac{\dot{v}_x(t)}{v_y(t)}\right] \end{cases} \tag{6}$$

and

$$|\omega(\mathrm{t})| = \sqrt{\omega_x^2(t) + \omega_y^2(t) + \omega_z^2(t)}\,. \tag{6a}$$

This formula (6) can be used in equations (1), (2) and (3) to transform them to the differentially flat system.

Note: If the values $\{v_x(t), v_y(t), v_z(t)\}$ in equations (6) are to appear as zero in any time instant, then the appropriate higher order derivatives of the aircraft's velocity should be used in relations (6) according to the transformation formula similar to de L'Hôspital rule.

An important property of flat systems is that we can find a set of outputs (equal in number to the number of inputs) so that we can express all states and inputs in terms of those outputs and their derivatives.[6] As results from equations (1), (2), (3) and (6) the arbitrary trajectory of an aircraft's movement can be easily identifiable in such a case, if and only if the multiple time differentiation of aircraft position readings can be realizable with sufficient precision. However, the system (1), (2), (3) and (6) is continuous whereas the readings are discrete in time.

STATEMENT OF THE IDENTIFICATION PROBLEM

The following question needs answering:

- how to estimate the continuous and smooth trajectory of an aircraft in order to derive its vector field of velocity and its higher order derivatives for prediction purposes (of the aircraft's future positions) based on the ground radar plots of past position readings (i.e. adequate time series of data) shifted on-line to the three dimensional sliding window?

To answer this question let us consider the possibility of numerical differentiation (sufficiently exact) of the smooth approximation of an arbitrary function $y = f(t)$ depending parametrically on time only.

Suppose that in discrete time instants $\{t_i;\ i = 0,1,2, ..., n\}$ equispaced with separation Δt in the finite period of time $[O,T]$, the output variable y was measured and took the discrete values $y_i \in [f(0), f(T)]$ for $i = 0,1,2, ..., n$ and furthermore the errors of measurements are considered inessential in this procedure. Now, having the set of data points $\{t_i, y_i;\ i = 0,1,2, ..., n\}$ find the approximation $y^* = f^*(t)$ for a real unknown continuous function $y = f(t)$ in such a way that the fitting $y^* = f^*(t)$ would be close to $y = f(t)$ in the sense of L_1 norm both for function and its derivative.

This aim of function approximation may be equivalently expressed by means of the following set of constraining conditions:

$$\begin{cases} y_i^* = f^*(t_i) \approx f(t_i) = y_i \\ \varphi^*(t_i) = \varphi_i^* = \dfrac{df^*(\tau)}{d\tau}_{|\tau=t_i} \approx \dfrac{df(\tau)}{d\tau}_{|\tau=t_i} = \varphi(t_i) \end{cases} \tag{7}$$

for $i = 0,1,2, ..., n$

and the solution to the task can be found by means of minimization of performance index Q, defined as below:

$$\min_{f^*} Q = \min_{f^*} \int_0^T \left\{ \left| f(\tau) - f^*(\tau) \right| + \right. \\ \left. + \left| \frac{d}{d\tau} f(\tau) - \frac{d}{d\tau} f^*(\tau) \right| \right\} d\tau \tag{7a}$$

where: $f^*(\tau)$ belongs to the class C^1 $(0,T)$ of continuously differentiable functions.

Note that the performance index (7a) is not differentiable in its global minimum and the solution to this problem could not be obtained from a variational principle. However, if the following boundary conditions $\{f^*(0)=f(0)\}$and $\{f^*(T)=f(T)\}$are fulfilled, then the performance index Q is non-negatively defined for all arbitrarily selected functions $f^*(\tau)\approx f(\tau)$ for $\tau\in(0,T)$ and the following relation is always valid: $Q\geq 0$.

Consequently, the minimal value of Q is to be zero, if and only if $f^*(\tau)\equiv f(\tau)$ for $\tau\in[0,T]$. This trivial statement means that:

- each smooth function $y=f(t)$ is the best approximation for itself in the finite domain $t\in[0,T]$.

Nevertheless, this observation does not indicate what form of the function $y=f(t)$ really is. It leads, however, to the fundamental conclusion that the structure of optimal solution $f^*(\tau)$ must be consistent with the „one-to-one" transformation and the approximating procedure should define diffeomorphism. Since the particular form of function $y=f(t)$ is actually unknown, all the efforts should be focused on the suitable fitting of the function $f^*(t)$ to the experimental data, in order to achieve the needed precision (i.e. $Q\leq\varepsilon$) with preferably the lowest cost of operation (i.e. simple computation, saving the computing time and memory resources, as well as assigning the robustness to noise). This nontrivial problem was first solved in 1972 by A. Korgul by inverted approximation originally developed for modeling and simulation of sugar extraction.[12, 13, 14]

SOLUTION TO THE IDENTIFICATION PROBLEM

Let us note that each continuous function $f(t)$ may be considered as a solution to a related but unknown differential equation. Thus, for each final and bounded interval of support $t\in[0,T]$ we can write:

$$f(t)=\int_0^t\frac{d}{d\tau}f(\tau)d\tau+f(0)$$

or **(8)**

$$f(t)=f(T)-\int_t^T\frac{d}{d\tau}f(\tau)d\tau$$

The relations (8) can also be combined to express function $y^*=f^*(t)$ with the skew-symmetric formula of unitary integration, as follows:

$$y_i^*=f^*(t_i)=\frac{1}{2}\left[f(0)+f(T)\right]+\frac{1}{2}\left\{\int_0^{t_i}\frac{d}{d\tau}f^*(\tau)d\tau-\right.$$
$$\left.-\int_{t_i}^T\frac{d}{d\tau}f^*(\tau)d\tau\right\}\qquad\textbf{(9)}$$

for $i=0,1,2,\ldots,n$ and $\tau\in[0,T]$

where:

the unknown derivative $\frac{d}{d\tau}f^*(\tau)=\varphi^*(\tau)$ is approached with the local basis functions $P_j(\tau)$ according to formula:

$$\varphi^*(\tau)=\sum_{j=0}^n\varphi_j^*\cdot P_j(\tau)\ \text{ for }\ j=0,1,2,\ldots,n\qquad\textbf{(10)}$$

The local basis functions $P_j(\tau)$ for this purpose are recommended in the form of piecewise square Lagrange polynomials.

Assuming additionally that the value of derivative φ_0^* at the left boundary point is defined in the form of:

$$\varphi_0^*=\frac{f(-\Delta t)-f(0)}{\Delta t}$$

or better **(11)**

$$\varphi_0^*=\frac{f(-\Delta t)-f^*(\Delta t)}{2\Delta t}$$

where $\Delta t=\frac{T}{n}$ and n is the even natural number and $f(-\Delta t)$ is an additional output variable obtained as the retarded value of previously measured output $f(0)$.

Substituting (10) and (11) for (9) we derive the following matrix equation:

$$\left[2\underline{y}^*-\underline{f}(0)-\underline{f}(T)-\varphi_0^*\cdot\underline{a}_0\right]_{n\times1}=[\mathbb{A}-\mathbb{B}]_{n\times n}\cdot\left[\underline{\varphi}^*\right]_{n\times1}\textbf{(12)}$$

where:

$$\underline{y}^*=\left[y_1^*,y_2^*,\ldots,f(T)\right]^T$$

$$\underline{f}(0)=\left[f(0),f(0),\ldots,f(0)\right]^T$$

$$\underline{f}(T)=\left[f(T),f(T),\ldots,f(T)\right]^T$$

$$\underline{a}_0=\left[a_{10},a_{20},\ldots,a_{20}\right]^T;\quad\underline{\varphi}^*=\left[\varphi_1^*,\varphi_2^*,\ldots,\varphi_n^*\right]^T$$

$$a_{10}=\int_0^{t_1}P_0(\tau)d\tau-\int_{t_1}^{t_2}P_0(\tau)d\tau;\quad a_{20}=\int_0^{t_2}P_0(\tau)d\tau$$

$$a_{ij}=\int_0^{t_i}P_j(\tau)d\tau;\quad b_{ij}=\int_{t_i}^{T}P_j(\tau)d\tau\quad\text{for }\ i,j=1,2,\ldots,n$$

For the proposed parabolical basis functions $P_j(\tau)$, the matrix $[\mathbb{A}-\mathbb{B}]$ is non-singular (i.e. det $[\mathbb{A}-\mathbb{B}]\neq 0$) and we can invert the equation (12) thus obtaining:

$$\begin{cases} \varphi_0^* = \dfrac{f(-\Delta t)-f(0)}{\Delta t} \text{ or } \varphi_0^* = \dfrac{f(-\Delta t)-f^*(\Delta t)}{2\Delta t} \\ \left[\underline{\varphi}^*\right]_{(n\times 1)} = [\mathbb{C}]_{(n\times n)} \cdot \left[2\underline{y}^* - \underline{f}(0) - \underline{f}(T) - \varphi_0^* \cdot \underline{a}_0\right]_{(n\times 1)} \end{cases} \tag{13}$$

where: $[\mathbb{C}]_{(n\times n)} = [\mathbb{A} - \mathbb{B}]_{n\times n}^{-1}$ and φ_0^* may also be defined in a different way, more sophisticated and more useful for particular purposes.

Equation (13) discloses how to approximately differentiate the unknown function $y = f(t)$, because it defines the values of derivatives φ_j^* of approaching function $y^* = f^*(t)$ in the nodal points t_i, $i = 0,1,2,...,n$. Note that this result seems to be essential for the philosophy of quantum computation, because it can accept as the input not one number but a coherent superposition of many different numbers and subsequently perform a computation (a sequence of unitary operations) on all of these numbers simultaneously. This can be viewed as a massive parallel computation.[16, 17, 18]

ERROR ESTIMATION

The validity of the presented procedure depends on the approximation accuracy. To estimate the approximation error, we transform relation (9), as follows:

$$2y_i - f(0) - f(T) = \sum_{j=0}^{n} \varphi_j^* \left[\int_0^{t_i} P_j(\tau) d\tau - \int_{t_i}^{T} P_j(\tau) d\tau \right] + R_i$$

where: **(14)**

$$R_i = \int_0^{t_i} \left[\varphi(\tau) - \sum_{j=0}^{n} \varphi_j^* \cdot P_j(\tau) \right] d\tau - \int_{t_i}^{T} \left[\varphi(\tau) - \sum_{j=0}^{n} \varphi_j^* \cdot P_j(\tau) \right] d\tau \tag{14a}$$

For equation (14a) we have estimation:

$$|R_i| \le \int_0^{T} \left| \varphi(\tau) - \sum_{j=0}^{n} \varphi_j^* \cdot P_j(\tau) \right| d\tau \tag{15}$$

Since the basis functions $P_j(\tau)$ are selected as parabola curves, the Simpson scheme can be used to estimate errors, thus giving the result:

$$|R_i| \le \frac{M_5 \cdot T \cdot \Delta t^4}{180} \tag{16}$$

where:

$$M_5 = \max_{[0,T]} \left| \frac{d^5}{d\tau^5} f(\tau) \right|; \qquad \Delta t = \frac{T}{n}.$$

Having estimation (16) and using equation (14a) we derive the following formula for absolute accuracy of approximation for derivatives:

$$\left| \frac{dy}{d\tau}\bigg|_{\tau=t_i} - \varphi_i^* \right| \le \frac{|R_i|}{\sum_{j=0}^{n} \left| a_{ij} - b_{ij} \right|} \tag{17}$$

and for unknown function:

$$\left| y_i - f_i^* \right| \le \frac{|R_i|}{2} \quad \text{for } i = 0,1,2,...,n. \tag{18}$$

Accordingly, the proposed method of approximation offers a globally limited error, both for fitted function and its time derivative. This error depends on the grain of time discretization as well as on the kind of local basis functions being used for fitting the time derivatives of output variable. If the fitted function $y = f(t)$ is to be a polynomial up to the fourth degree, the presented method is exact, as $\{M_5 \equiv 0\}$.

The power of convergence, the accuracy of setting and the speed of setting are guaranteed, as the absolute approximation error of the fitted function and its derivative are globally limited according to (16), (17) and (18).

In conclusion, we can insist, that the proposed method is precise enough to differentiate an arbitrary unknown time series of data and it can be easily used for trajectory tracking purposes.

APPLICATION OF TDNN FOR TRAJECTORY RECOGNITION

In position where the popular tracking techniques are well suited for recognition of the straight line or moderately curved trajectories only, and the modeling of the aircraft's high maneuvers needs to use bank filters or more sophisticated alpha-beta-gamma trackers or even filters with the switching of the structure,[2, 3, 4] the analysis and design methods based upon knowledge about phase-space dynamics of nonlinear and dissipative chaotic systems seem to be the most promising tool for this purpose.[10] Thus the time series of radar plots can be treated rather as a chaotic time series of measured data, where the real unknown value is disturbed with the equivalent non-Gaussian or non-Markovian stochastic process (e.g. including also spikes and outliers). In this case, the full state vector of a chaotic dissipative system can be easily reconstructed and tracked using only the number of the past position readings of the scalar components (coordinates) of the observed state variables (dependent parametrically on time only). For this purpose one should note that each and unknown trajectory of an aircraft should be considered as a stable attractor of the dissipative process of aircraft flight in aerodynamic environment. In recent years, considerable progress has been made in modeling chaotic time series with neural networks. Though neural networks are not a panacea, they can provide a significant advantage over classic techniques and can also be implemented for real-time solutions.[11]

Using now the Takens theorem we can see that for aircraft trajectory recognition (or even its prediction) with the neural network it is enough to know $(2m + 1)$ past position readings.[19] In our case the number m relates simply to the quantity of state equations (2) and reads four. Now we can develop the moving window in the form of generalized „tapped delay line" and organize it as a linear memory structure (comprising eight memory cells only) with the current data shifting. This window is supplied serially on-line with the data of succeeding position readings of aircraft, and the data from this window (i.e. the whole stream of memorized data) are parallely supplied to the inputs of perceptron structure (see Fig.2).

The perceptron is organized according to equation (13). After its convenient transformation the time differentiation of each coordinate (x, y and z) takes the form of the following formula:

$$\begin{bmatrix} \dot f(t-6\Delta t) \\ \dot f(t-5\Delta t) \\ \dot f(t-4\Delta t) \\ \dot f(t-3\Delta t) \\ \dot f(t-2\Delta t) \\ \dot f(t-\Delta t) \\ \dot f(t) \end{bmatrix} = \frac{1}{\Delta t}[\mathbb{E}]\cdot[\mathbb{D}] \begin{bmatrix} f(t-8\Delta t) \\ f(t-7\Delta t) \\ f(t-6\Delta t) \\ f(t-5\Delta t) \\ f(t-4\Delta t) \\ f(t-3\Delta t) \\ f(t-2\Delta t) \\ f(t-\Delta t) \\ f(t) \end{bmatrix} \qquad \textbf{(19)}$$

In equation (19) matrix $[\mathbb{E}]$ takes the form:

$$[\mathbb{E}] = \begin{bmatrix} e_{00} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_{11} & e_{12} & e_{13} & e_{14} & e_{15} & e_{16} \\ 0 & e_{21} & e_{22} & e_{23} & e_{24} & e_{25} & e_{26} \\ 0 & e_{31} & e_{32} & e_{33} & e_{34} & e_{35} & e_{36} \\ 0 & e_{41} & e_{42} & e_{43} & e_{44} & e_{45} & e_{46} \\ 0 & e_{51} & e_{52} & e_{53} & e_{54} & e_{55} & e_{56} \\ 0 & e_{61} & e_{62} & e_{63} & e_{64} & e_{65} & e_{66} \end{bmatrix} \qquad \textbf{(19a)}$$

and matrix $[\mathbb{D}]$ takes the form:

$$[\mathbb{D}] = \begin{bmatrix} d_{01} & d_{02} & d_{03} & d_{04} & 0 & 0 & 0 & 0 & 0 \\ d_{11} & d_{12} & d_{13} & d_{14} & 0 & 0 & 0 & 0 & d_{19} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} & 0 & 0 & 0 & d_{29} \\ d_{31} & d_{32} & d_{33} & d_{34} & 0 & d_{36} & 0 & 0 & d_{39} \\ d_{41} & d_{42} & d_{43} & d_{44} & 0 & 0 & d_{47} & 0 & d_{49} \\ d_{51} & d_{52} & d_{53} & d_{54} & 0 & 0 & 0 & d_{58} & d_{59} \\ d_{61} & d_{62} & d_{63} & d_{64} & 0 & 0 & 0 & 0 & d_{69} \end{bmatrix} \textbf{(19b)}$$

Note: If the higher order derivatives were needed (e.g. $\ddot f$) the additional values of $\dot f(t-8\Delta t)$ and $\dot f(t-7\Delta t)$ should be defined as an appropriately retarded value of $\dot f(t-6\Delta t)$ to adapt the formula (19) for further computations.

The moving window and adjacent perceptron structure for differentiation of measured data can be presented schematically by means of the graph of time delay neural network (TDNN), as outlined on the Fig. 2.

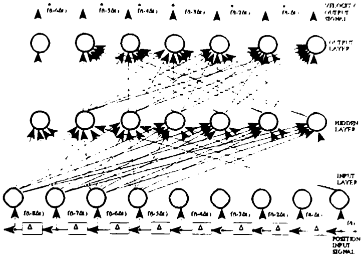


Fig. 2. Time delay neural network for differentiating radar plots (one-dimensional case only)

This structure was successfully used for on-line derivation of the following vector fields along an identified trajectory:

- linear velocity of aircraft $\{v_x(t), v_y(t), v_z(t)\}$
- linear acceleration of aircraft $\{\dot v_x(t), \dot v_y(t), \dot v_z(t)\}$
- jerk of aircraft $\{\ddot v_x(t), \ddot v_y(t), \ddot v_z(t)\}$
- angular rate of aircraft $\{\omega_x(t), \omega_y(t), \omega_z(t)\}$
- angular acceleration of aircraft $\{\dot\omega_x(t), \dot\omega_y(t), \dot\omega_z(t)\}$

The on-line identification of the above vector fields allows to recognize the arbitrary trajectory of the maneuvering aircraft and, using equations (1), (2), (3) and (6), to predict precisely enough near future positions. This fact was also used for trajectory generation during simulation experiment developed to prove the usefulness of this methodology.

SIMULATION EXPERIMENT

Numerical simulation was used to prove the proposed method of trajectory recognition. For this purpose, the pursuit scenario was modeled. The time series of numerical data was artificially generated by means of kinematic equations (1), (2) and (3) to imitate the successive measurements of aircraft position in 3-D space. These data were tracked on-line according to procedure (19) which was used as a basic task in target interception modeling.[15] Using this model of target interception, several simulation experiments were made and the following results were obtained.

The four pursuit scenarios were simulated numerically and are shown graphically on (x,y) plane of Figs. 3, 4, 5 and 6 respectively.

Fig. 3 presents maneuvering target interception according to long-range proportional navigation guidance.

A similar airborne situation is illustrated in Fig. 4 but for different initial values of homing process.

In Fig. 5 the tail-chase is presented, whereas in Fig. 6 the frontal attack is illustrated.

The related numerical data characterizing these scenaries are specified in the Table 1.

Table 1. Initial and final state conditions of interception

No	v_c	$\max v_{m_1}$	$\min v_{m_1}$	$\max v_{m_2}$	$\min v_{m_1}$	μ_m	ψ_m	d_{rad}	d_r^m	t_z	t_k	$t_k - t_z$	x_c^k	y_c^k
	$m \cdot sec^{-1}$	$m \cdot sec^{-1}$	$m \cdot sec^{-1}$	$m \cdot sec^{-1}$	$m \cdot sec^{-1}$	-	*deg*	*m*	*m*	*sec*	*sec*	*sec*	*km*	*km*
1.	200	200	100	300	100	0,5	60	30000	5000	200	930	730	71,3	69,2
2.	260	259	207	400	207	0,5	0	30000	5000	500	1030	530	137,2	119,5
3.	200	300	300	300	100	0,5	0	30000	5000	300	800	500	49,3	55,5
4.	260	259	207	400	207	0,7	60	30000	5000	300	340	340	83,3	61,2

These examples of numerical simulation evidently indicate that the TDNN procedure developed for the differentiation of time series of succeeding aircraft positions worked sufficiently well (without amplification of numerical errors due to discretization and quantization of input data). It can be successfully used for efficient realization of numerical procedures for aircraft trajectory tracking and target interception even against to presence of the round off errors generated during data processing using computer equipment (due to finite word length).

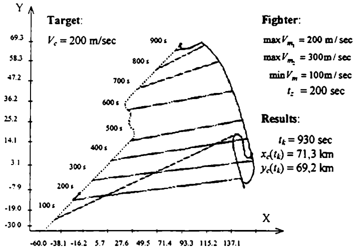


Fig. 3. Target - interception according to long-range proportional navigation guidance

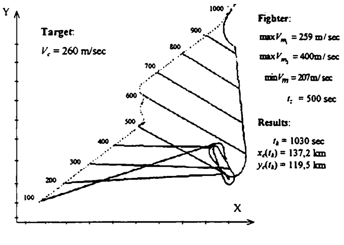


Fig. 4. Target - interception according to long-range proportional navigation guidance

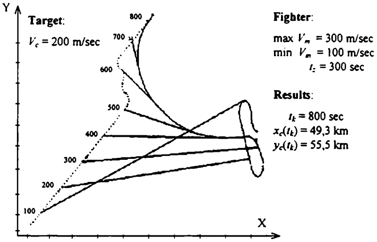


Fig. 5. Target - interception according to tail-chase on (x,y) plane

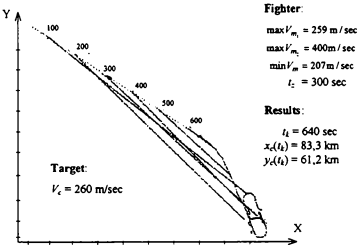


Fig. 6. Target - interception according to frontal attack on (x,y) plane

It was possible since both aircraft positions and their several time derivatives were smoothly reconstructed (on the basis of their discrete readings) according to the parallel procedure of diffeomorphical approximation (in the sense of least absolute deviation i.e. L_1 norm). Because the numerical differentiation was developed simply as an algebraic inversion of the symplectic integration, high-speed computation, robustness to noises (errors) and low-cost reconstruction of signals were assigned.

All these aspects of numerical integration of differential equations by symmetric composition methods have been also considered lately in physics.[20, 21, 22, 23] But it seems that proposed TDNN structure can be rather fruitfully exploited in real-time applications, if this scheme were considerably simplified and realized in VLSI technology. The form of the bank of differentiating filters for recognizing various classes of signals (not only the trajectory) is suggested.

CONCLUSIONS

A coordinated high rate turn from modern tactical aircraft or missiles cannot be approximated by a ramp motion, by random white noise acceleration or by random jerk motion. For these dynamic maneuvering tracks the classic theories lead to highly biased or incorrect solutions. A full set of nonlinear kinematic equations combined with flow equations of velocity vector field of the maneuvering aircraft was used to formulate the original nonlinear tracking problem for a time-varying flat system. The time delay neural network (TDNN) was proposed for the differentiation and smoothing of an aircraft's 3-D positions in order to identify the translational and rotational movements of aircraft. In this proposition the differentiation was realized as an algebraic inversion of symplectic integration. Thereupon it guarantees inherent smoothing of the data contaminated with the noise (errors). The proposed TDNN structure consists of an universal approximator for smooth function. Its main advantage includes avoiding the curse of dimensionality and the local minima problem. This structure is easy to apply in real-time implementations and it can be realized in VLSI technology.

ACKNOWLEDGEMENTS

Although the Polish AK method of differentiation was theoretically elaborated in 1972, its first application (1978-1980) for solving the hyperbolic partial differential equations could not be possible without the support of Prof. J. Pułaczewski and Dr. B. Beliczyński at the University of Technology in Warsaw. I am very grateful to my colleagues R. Bedyński, M.Sc. and K. Hołubowicz, M.Sc. for their help in programming work provided during this first development. The solving of the target interception (with the use of this method of differentiation) was developed under the direction of Prof. E. Kołodziński at the Institute of Automation of Command Systems at the Military University of Technology in 1991-1994. I would like to thank Dr. E. Kowalska, Dr. T. Pietkiewicz and M.Sc. K. Płomiński for their support in programming work, as well as several other of my friends for helpful discussions and input to the final form of this paper.

REFERENCES

[1] Mazor E., Averbuch A., Bar-Shalom Y. and Dayan J.: „Interacting Multiple Model Methods in Target Tracking: A Survey", *IEEE Transactions on Aerospace and Electronic Systems*, Vol.34, No. 1, 1998, pp.103-123

[2] Wheaton B.J., Maybeck P.S.: „Second-Order Acceleration Models for an MMAE Target Tracker", *IEEE Transactions on Aerospace and Electronic Systems*, Vol.31, No. 1, 1995, pp.151-166

[3] Mehrotra K., Mahapatra P.R.: „A Jerk Model for Tracking Highly Maneuvering Targets", *IEEE Transactions on Aerospace and Electronic Systems*, Vol.33, No. 4, 1997, pp.1094-1105

[4] Blair W.W.: „Fixed-Gain Two-Stage Estimators for Tracking Maneuvering Targets", *IEEE Transactions on Aerospace and Electronic Systems*, Vol.29, No. 3, 1993, pp.1004-1014

[5] Wang T.C., Varshney P.K.: „A Tracking Algorithm for Maneuvering Targets", *IEEE Transactions on Aerospace and Electronic Systems*, Vol.29, No. 3, 1993, pp.910-924

[6] Van Nieuwstadt M.J., Murray R.M.: „Rapid Hover-to-Forward-Flight Transitions for a Thrust--Vectored Aircraft", *Journal of Guidance, Control, and Dynamics*, Vol.21, No. 1, 1998, pp.93-100

[7] Kaminer I., Pascoal A., Hallberg E. and Silvestre C.: „Trajectory Tracking for Autonomous Vehicles: An Integrated Approach to Guidance and Control", *Journal of Guidance, Control, and Dynamics*, Vol.21, No. 1, 1998, pp.29-38

[8] Arnold V.I.: *Mathematical Methods of Classical Mechanics*, Springer-Verlag, New York, NY, 1989

[9] Huang T.S.: „Visual Motion Analysis" in: *Encyclopedia of Artificial Intelligence*, ed. by S.C. Shapiro, Wiley, 1992, pp.1638-1654

[10] Hrycej T.: „Error Measure for Identifying Chaotic Attractors", *Proceedings of the 5th European Symposium on Artificial Neural Networks, ESANN '97*, Bruges (Belgium), 1997, pp.139-144

[11] Rodin E.Y., Amin S.M.: „Maneuver Prediction in Air Combat via Artificial Neural Networks", *Computer Math. Applic.* Vol.24, No. 3,1992, pp.95-112

[12] Korgul A.: „The New Computing Method for Countercurrent Exchange Phenomena Simulation", *Proceedings of the International AMSE Conference on Modeling and Simulation*, Sorrento (Italy), 1986, Vol.3.4, pp.41-54

[13] Korgul A.: „Mathematical Modeling and Simulation of the Sugar Extracting Process", *Zuckerindustrie*, Vol.113, No. 4, 1988, pp.304-309

[14] Korgul A.: „Networks for Learning and Differentiating an Input-Output Mapping", *Proceedings of the International Conference on Artificial Neural Networks, ICANN'93*, Amsterdam (The Netherlands), 1993, p.475, Springer-Verlag, London

[15] Korgul A.: „Time Delay Neural Network for Target-Intercept Problem Solving", *Proceedings of the 5th European Symposium on Artificial Neural Networks, ESANN'97*, Bruges (Belgium), 1997, pp.339-344

[16] Korgul A.: „The Method and Device for Recognizing of the Smooth Signals Especially in Electronics", *Patent application P-310026* filled at 16. August, 1995 in Polish Patent Office

[17] Spiller T.P.: „Quantum Information Processing: Cryptography, Computation, and Teleportation", *Proceedings of the IEEE*, Vol.84, No. 12, 1996, pp.1719-1740

[18] Ekert A.: „Quantum Cryptoanalysis - Introduction", http://eve.physics.ox.ac.uk/QCresearch/ /cryptoanalysis/qc.html

[19] Takens F.: „Detecting Strange Attractors in Turbulence", *Lecture Notes in Mathematics*, Vol.898 (Warwick 1980), pp.366-381, Springer-Verlag, Berlin

[20] Forest E., Ruth R.D.: „Fourth-Order Symplectic Integration", *Physica D 43*, 1990, pp.105-117

[21] Yoshida H.: „Construction of Higher Order Symplectic Integration", *Physics Letters A*, Vol.150, No. 5,6,7, 1990, pp.262-269

[22] McLachlan R.I.: „On the Numerical Integration of Ordinary Differential Equations by Symmetric Composition Methods", *SIAM J.Sci. Comput.*, Vol.16, No. 1, 1995, pp.151-168

[23] Moreau Y., Vandewalle J.: „Composition Methods for the Integration of Dynamical Neural Networks", *Proceedings of the 5th European Symposium on Artificial Neural Networks, ESANN'97*, Bruges (Belgium), 1997, pp.303-308

AIAA-98-4165

SENSITIVITY OF AIR COMBAT MANEUVERS TO AIRCRAFT-MODELING UNCERTAINTIES

Jaakko Hoffren* and Jari Vilenius†
Helsinki University of Technology, Laboratory of Aerodynamics
P.O. Box 4400, FIN–02015 HUT, FINLAND

The sensitivity of predetermined maneuvers characteristic of close-in air combat has been studied to clarify the accuracy requirements for aircraft modeling. Nine dynamic maneuvers were calculated using an improved version of the AML-75 air combat simulator, where six parameters of a fighter model were systematically varied. Additional steady-state performance calculations were conducted to facilitate the generalization of the findings, according to which aircraft weight is the most important factor. Maximum lift, thrust and drag are also of importance, but roll performance is secondary.

1 Introduction

Air combat simulations can be applied, among other purposes, to develop efficient tactics against perceived threats. The studies require that representative numerical models of the aircraft involved are available. Often, however, such information related to foreign equipment is inaccessible, and models must be created based on scarce data. Owing to the problem set-up, only performance models of computer-piloted adversaries can be envisaged for the purpose. The creation of a threat model is a relatively laborious job, and it is not immediately evident if representative simulation results can be expected using the necessarily crude modeling techniques. To direct the modeling effort appropriately, the accuracy requirements for model parameters and their relative importance should be known. Since the model parameters themselves, like drag and thrust, do not reveal well the practical value of a model, the important quantities to evaluate are the flight paths and other performance figures predicted with the model in air combat situations.

To assess the feasibility of threat modeling and to establish accuracy requirements for model parameters, actual air combat simulations would appear to offer the most realistic test framework. However, they are not considered here due to the related difficulties of obtaining statistically representative results in such complex multi-disciplinary situations. For reliable conclusions, a massive test campaign involving various aircraft types and initial conditions applying random factors in decision-making would be required.

An alternative, more directly controlled approach to studying the modeling requirements is to perform a sensitivity analysis for a set of predetermined maneuvers that are assumed to be representative of an air combat situation. In this way, the pure effects of the model characteristics on aircraft dynamics are found, although the significance of the differences in an actual combat remains uncertain. In this paper, such a study using nine maneuvers selected using Ref. 1 is described. As the differences in dynamic maneuvers depend strongly on the aircraft weight, thrust and drag, additional steady-state performance evaluations are conducted to help to generalize the findings from the simulations into the whole Mach-altitude envelope. The results lead to clear conclusions of the modeling requirements, which are collected at the end of the paper.

2 Maneuver Simulations

2.1 Simulation Method

For the computational tool, an updated version of the published AML-75 air combat simulator[2,3] was applied. The basic version uses an aircraft model written into the code, and the flight path integration is based on a non-physical, two-mode procedure. These features had to be changed to obtain a useful simulation platform for the present study.

The modifications to enable the use of external aircraft

*Research Scientist, member AIAA
e-mail Jaakko.Hoffren@hut.fi

†Research Engineer, member AIAA
present e-mail Jari.Vilenius@finnair.fi.

data files are trivial, but the enhancement of the flight path calculation warrants some discussion. With a performance model, the calculation does not use the moment equations of motion at all, but applies some limiting of the angular motions to maintain reasonable physical behaviour. The control parameters defining the derirable motion are the throttle setting δ_T, load factor n_z and bank angle ϕ. The first two parameters come directly from the decision-making logic, replaced in this study by commands to obtain the specific maneuvers, but the bank angle follows from the desired maneuver plane rotation angle ρ_{mp} from the vertical as

$$\phi = \rho_{mp} - arcsin(\frac{Wcos\bar{\theta}sin\rho_{mp}}{L + Tsin\alpha}) \tag{1}$$

where W, L and T are the aircraft weight, lift and thrust, $\bar{\theta}$ is the velocity vector pitch angle and α the angle of attack. According to the definition of ρ_{mp}, the aircraft moves in the maneuver plane that contains the force resultant.

In the original AML-75, drag, lift and thrust are kept constant during a transition phase from one maneuver plane to another, which may take several seconds. The transition is always computed to take place at the maximum steady-state roll rate, and the inevitable nonphysical motion is afterwards smoothed by a filter for display. Mostly because of these shortcomings, the flight path calculation routine was replaced by a completely new procedure that continuously tries to change the aircraft state towards the commanded $(\delta_T, n_z, \rho_{mp})$ combination applying a single-degree-of-freedom dynamics for the angular motions and thrust response.

For example, in the first step to change the bank angle ϕ at each time step towards the commanded value ϕ_{comm} obtained from Eq. (1), an extrapolated bank angle

$$\phi_{ext} = \phi_t + p_t \Delta t_{pred} \tag{2}$$

is calculated using the current bank angle ϕ_t, roll rate p_t and a prediction time interval Δt_{pred}, chosen as half the roll time constant. The desired roll rate p_{comm} depends on the difference of the commanded and extrapolated bank angle as

$$|p_{comm}| = min[|k(\phi_{comm} - \phi_{ext})|, p_{max}(H, Ma, \alpha)] \tag{3}$$

Here the gain k is selected as $p_{max}/20°$, giving a maximum possible command when the difference of angles is greater than 20°. The maximum steady-state roll rate $p_{max}(H, Ma, \alpha)$, depending on the altitude H, Mach number Ma and the angle of attack is characteristic to each aircraft type and configuration. The direction of the roll command follows from the difference $\phi_{comm} - \phi_{ext}$ that is always reduced to less than 180°. The updated roll rate $p_{t+\Delta t}$ can now be calculated from

$$p_{t+\Delta t} = p_t + \frac{1}{\tau_p}(p_{comm} - p_t)\Delta t \tag{4}$$

where the roll time constant τ_p determines the angular acceleration, and Δt is the step of the time integration, chosen as 0.01 seconds in this study. Finally, the updated bank angle follows from an equation of the form (2) using the updated roll rate and Δt.

Test calculations with the new flight path calculation method have validated its realism, at least in simple, well-defined maneuvers. It is felt that in this respect, the updated code, called AML-75/HUT, probably exceeds the realism of the latest commercial performance-model-based AML's.

2.2 Aircraft Model

The aircraft model utilized by AML-75/HUT consists of numerical data arrays describing the necessary characteristics as functions of typically two arguments, like the maximum thrust as a function of altitude and Mach number, or the drag coefficient as a function of lift coefficient and Mach number. During the execution, appropriate values are interpolated from these arrays by applying a multi-dimensional Hermite interpolation[4] that ensures a continuous slope of the function surface.

As the test aircraft, the F-16A was selected in this study as it represents a current-generation high-performance fighter. The aircraft is assumed to carry only the wing-tip missiles as external stores, and its mass is 11 000 kg, a value representative of air combat. However, because the aircraft model was built using published data of the aircraft type and relatively simple modeling methods, the results are more of a generic nature than specifically related to the F-16.

For example, the thrust values for an F-100 engine at the military setting and at maximum afterburner in the whole flight envelope were predicted on the basis of the published static thrusts at ISA, sea-level conditions for an uninstalled engine and of the main engine parameters, like by-pass ratio, overall pressure ratio and turbine inlet temperature.[5] For the predictions, a PC code, called GASTURB,[6] was applied. In the drag estimation, DATCOM[7] and Raymer's book[8] were among the various sources utilized, in addition to some statistical data. The maneuvering characteristics were estimated entirely based on empirical knowledge.

The modeling approach is documented in more detail in Ref. 9. Although the model may not be accurate for a specific aircraft type, it is representative enough for valid conclusions of the sensitivity analysis under consideration.

2.3 Studied Maneuvers

Level Acceleration. The maneuvers studied were aimed to be representative of a close-in air combat. Different types of maneuvers, selected using Ref. 1, were considered to test various aspects of the aircraft model. All the maneuvers were initiated in level flight at an altitude of 5000 meters, and the aircraft weight was kept constant during the runs. The results given in this subsection correspond to the unmodified baseline model and serve as references for the results obtained with model variations.

As the first case, a level acceleration with maximum afterburner starting at $Ma = 0.4$ was studied to integrate the effect of specific excess power over a relatively broad speed range. In this maneuver, the flight path is just a straight line, and the only important parameter is the Mach number, which increased to 1.26 during 1 minute of simulation.

Level Turns. Different turns in a roughly horizontal plane are typical elements of an air combat.[1] Here, three idealized situations involving a 360° level turn using maximum afterburner were studied. The turns were initiated from a straight flight one second after the beginning of the simulation, and the wings were leveled after a full circle to continue in the original direction.

In the first turn, the initial Mach number was 0.90, and the load factor of 5.74 was selected to correspond to a steady state so that subsequent model variations become easily visible. Some important state variables as functions of time and the flight path seen from above are shown in Fig. 1. The small, visible adjustments of the bank angle are required to keep altitude constant.

In the second turn, the initial conditions were identical to the first one, but the commanded load factor was 7.0 to obtain a better turn rate. As seen in Fig. 2, the command leads to a tightening orbit at a decreasing velocity, which forces the load factor to be relaxed in the latter part of the turn to maintain the angle of attack within limits. The case tests the aircraft maneuverability near the corner speed.

The third turn took place almost entirely supersonically to test high-speed capabilities and for comparisons with the subsonic turns. The initial Mach number was 1.4, and the load factor was kept constant at 7.0. At the end of the resulting, tightening turn lasting 34 seconds, the Mach number was reduced to 0.97.

Split-S. The fifth maneuver tested the ability to change direction in the vertical plane at low speeds. Initially, the aircraft was in steady wings-level flight at $Ma = 0.4$. The 180° direction change was accomplished by a maximum-effort half roll, followed immediately by a half loop at the lift boundary, as illustrated in Fig. 3. Again, the thrust was set to maximum afterburner for the duration of the maneuver.

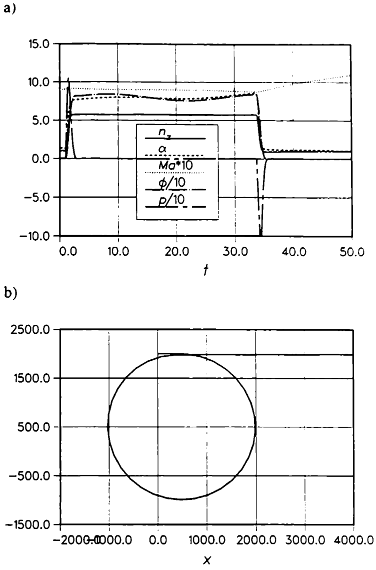


Fig. 1: a) Most important state variables as functions of time b) flight path seen from above during a steady-state level turn at $Ma = 0.90$ with a generic F-16.

Flat Scissors. An aerial duel sometimes tends to develop into a series of successive roughly level turns in alternating directions, when the combatants try to get behind each other.[1] The maneuver is generally known as flat scissors, where the velocities are typically fairly low and decreasing.

The maneuver, initiated here at $Ma = 0.70$ with wings level, was simulated as a series of level turns, where the direction changes were predetermined to take place when the heading differed ±45° from the original heading. The load factor in the turns was specified to be 80 percent of the lift-limited maximum. The changes in direction were always done at maximum effort, and the load factor was relaxed during the rolls to improve roll rate and to prevent pull-ups. Maximum afterburner thrust was applied, and the altitude was held by small adjustments in the bank angle. During a minute of simulation, the aircraft performed six direction changes with a bleeding velocity, as seen in Fig. 4.

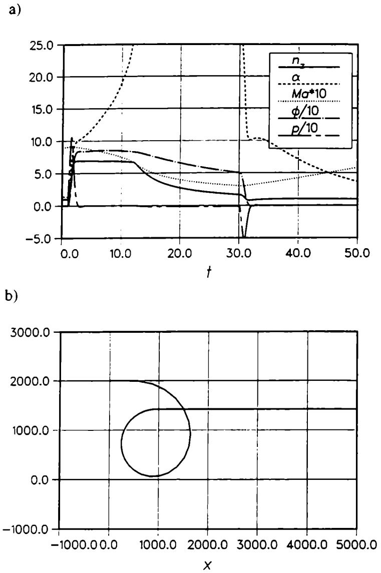


Fig. 2: a) Most important state variables as functions of time b) flight path seen from above during a commanded 7G level turn initiated at $Ma = 0.90$ with a generic F-16.

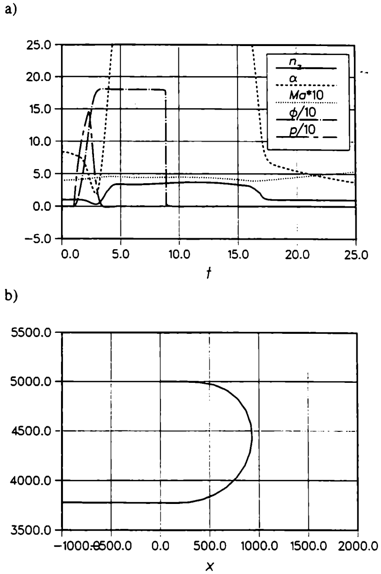


Fig. 3: a) Most important state variables as functions of time b) vertical projection of flight path during a split-S initiated at $Ma = 0.40$ with a generic F-16.

Rolling Scissors. Instead of the flat scissors, a duel may develop into successive barrel rolls, especially if the velocities are relatively high.[1] As the seventh case, this kind of a maneuver, known as rolling scissors, was studied. The situation was idealized into a series of predetermined barrel rolls, performed using maximum afterburner and initiated at $Ma = 0.90$. The load factor command was sought iteratively to vary between 3.73 and 6.53 during each 20-second roll in order to obtain reasonable flight paths in an averagely horizontal plane. The results for the nearly steady-state maneuvering are shown in Fig. 5.

Defensive Spiral. According to Ref. 1, a combatant in trouble may try to force the opponent behind him to overshoot by performing a steep spiral at low speed with idle thrust and the airbrake deployed. Such a situation is suitable for a test case to give contrast to the preceeding maneuvers involving high thrust and benefit from low drag.

The maneuver was initiated in level flight at $Ma =$ 0.70 by reducing the thrust to flight idle, opening the airbrake and rolling the aircraft to 90° of bank. The load factor command was 90 percent of the lift-limited maximum. During one minute, the aircraft performed slightly less than two full revolutions, lost 3400 meters in altitude and slowed down to a stable Mach number of 0.33. The load factor was reduced from the initial transient of about 6 to slightly over 2.

Turn Initiation. As the last test maneuver, a maximum-effort 8G turn initiation was studied. This was done to test specifically the rolling capability that plays a minor role in the preceeding long-lasting maneuvers. The Mach number was 0.90, and the bank angle was stabilized to correspond to the selected load factor. The maneuver took about 2 seconds, during which the maximum roll rate peaked at slightly over 100°/s.

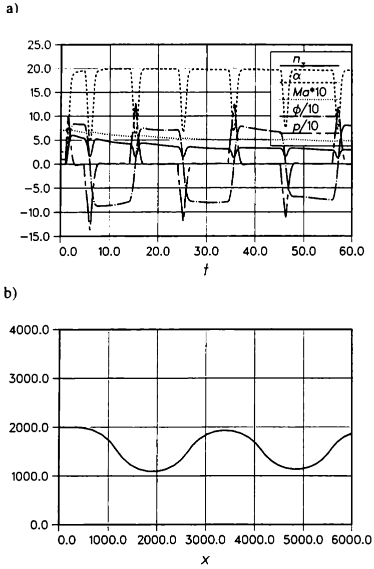


Fig. 4: a) Most important state variables as functions of time b) flight path seen from above during flat scissors initiated at $Ma = 0.70$ with a generic F-16.

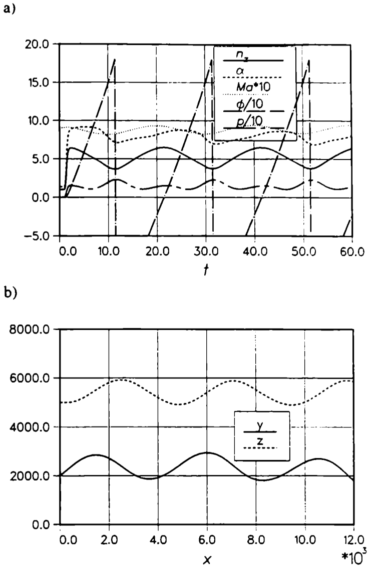


Fig. 5: a) Most important state variables as functions of time b) horizontal (y) and vertical (z) flight path projections during rolling scissors initiated at $Ma = 0.90$ with a generic F-16.

2.4 Model Variations

In the sensitivity analysis, six model parameters were varied. Variations were done in both directions from the nominal values to reveal possible strong nonlinearities, and the magnitudes of variations were selected to represent expected modeling uncertainties.

Although aircraft weight is not a parameter to be modeled, its value has a strong effect on the aircraft performance and must be known accurately. Here, variations of ±5 percent were considered to represent, for example, differences in fuel level.

The accuracy of the thrust and drag estimates can be assumed to be typically better than ±10 percent. Such variations from the instantaneous baseline value for these roughly equally important parameters were applied.

In some of the maneuvers studied, the maximum lift coefficient is of decisive importance. This parameter, difficult to estimate reliably, was varied ±10 percent.

In the lateral maneuvers, the roll performance is meaningful. Both the maximum steady-state roll rate and the roll time constant affect the capability, as seen in Eqs. (3,4). There may be large errors in the estimates for the roll rate and the time constant, and therefore variations of ±30 percent were applied for these parameters. For the time constant, large errors are unavoidable in practical calculations with basic AML-type programs because a single constant value must be specified for the whole flight envelope.

Because not all the variable parameters are of significance in all maneuvers, only cases with expected changes were calculated. The cases studied are defined in Table 1. Altogether, 81 test runs were conducted.

2.5 Sensitivity of Different Maneuvers

Level Acceleration. In an acceleration, thrust is higher than drag, which makes the sensitivity to the former the higher of the two. The difference in sensitivity depends on the speed range, but in the present case, the effect of

Table 1: Model variations tried in the test maneuvers

Test maneuver	Baseline model	Δm ±5%	ΔT ±10%	ΔD ±10%	$\Delta C_{L_{max}}$ ±10%	Δp_{max} ±30%	$\Delta \tau_p$ ±30%
Level acceleration	+	+	+	+			
Level turn, M0.9/5.74G	+	+	+	+		+	+
Level turn, M0.9/7.0G	+	+	+	+	+		
Level turn, M1.4/7.0G	+	+	+	+			
Split-S	+	+	+	+	+	+	+
Flat scissors	+	+	+	+	+	+	+
Rolling scissors	+	+	+	+			
Defensive spiral	+	+	+	+		+	+
Turn initiation	+					+	+

thrust variations was about three-fold to the effect of drag variations with no strong nonlinearities. When thrust was increased by 10 percent, the time to reach $Ma = 1.0$ decreased by 11 percent, and the final Mach number increased by 8 percent.

The sensitivity to weight variations lay mostly between the thrust and drag effects. At low speeds, a 5 percent weight variation is more important than a 10 percent drag change, but as speed builds up, the situation is reversed.

Level Turns. In the first turn ($Ma_{initial} = 0.90$, $n_z = 5.74$), thrust balances drag in the nominal conditions, which leads to similar sensitivities to these parameters. The time to complete a 360° turn increases by 3 percent and the final velocity by 6 percent when the thrust is increased or the drag decreased by 10 percent. On the other hand, a 10 percent thrust reduction or a similar drag increment shortens the time of turn by 6 percent and reduces the final velocity by 2 percent, which means that the effects are quite nonlinear.

Similar changes as above are brought about by ±5 percent variations in weight, making the sensitivity to the weight twice as high as to the thrust or drag. This follows from the increase in the induced drag with weight in addition to its direct effects. The results related to a 5 percent weight reduction are shown in Fig. 6 to enable comparisons with Fig. 1.

Because the aircraft rolls only at the turn initiation and at leveling off in the turn cases, the effect of roll performance is marginal, as expected. The 30 percent variations in roll parameters changed the time to complete the turn by less than 0.5 percent, and therefore, they were not considered in the other turns.

In the other subsonic turn with the commanded n_z of 7.0, the drag and the sensitivity to it are higher in relation to the thrust. A 10 percent drag reduction shortens the turn time by 5 percent and increases the final velocity by 20 percent. The sensitivity to the thrust is slightly more than half of that, and some nonlinear effects are evident.

Because the lift coefficient in the second turn is high, the effect of weight is even larger than in the first turn via the induced drag. The variations in weight produce qualitatively similar results as the drag variations, but the sensitivity magnitudes are more than doubled. The sensitivity to weight can be clearly seen by comparing Figs. 2 and 7.

In addition to the weight, the maximum lift coefficient is an important parameter in a turn below the corner speed. In the case under discussion, 10 percent changes in $C_{L_{max}}$ affect the turn time less than 5 per cent changes in weight, but with respect to the final velocity, things are reversed.

In the supersonic turn with a decreasing velocity, drag is the most powerful parameter. A drag reduction of 10 percent increases the turn time by 7 percent and the final Mach number by 16 percent. The thrust effects are roughly 60 percent of the drag effects, and the weight effects lie between these two.

It may appear confusing that the improvement in aircraft performance by a thrust increase or drag reduction can increase the turn time and radius. However, this is a result of the adopted control strategy defining a load factor. In real life, the performance improvement could be exploited by increasing the load factor instead of the final velocity at the completion of the maneuver, as done here.

Split-S. In the split-S maneuver, all the variables of the analysis are active, but at least the case studied was relatively insensitive to almost all of them.

The thrust and drag variations did not affect the flight path or the time to complete the maneuver appreciably. Instead, a 10 percent drag reduction increased the maximum load factor during the maneuver by 10 percent and the exit velocity by 7 percent. The thrust effects were roughly half of these figures.

A 5 percent weight reduction shortened the maneuver

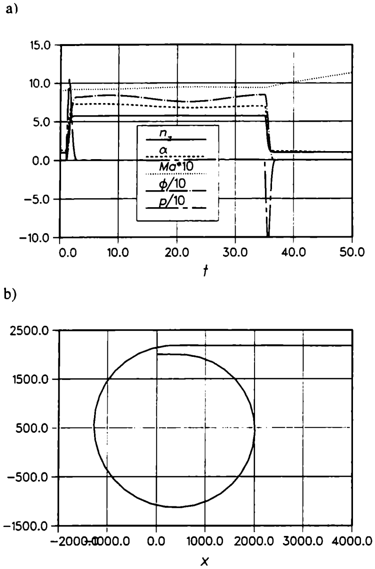


Fig. 6: a) Most important state variables as functions of time b) flight path seen from above during a 5.74G level turn initiated at $Ma = 0.90$ **with a generic F-16 at a 5 percent lowered weight.**

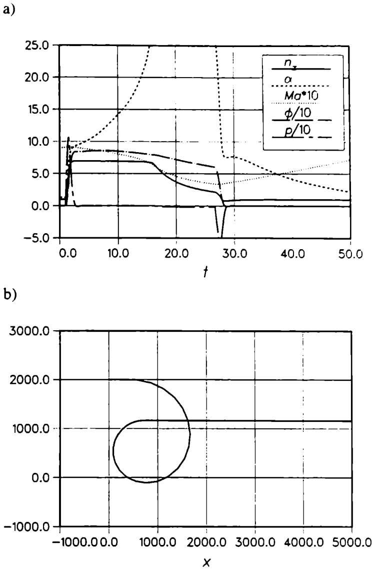


Fig. 7: a) Most important state variables as functions of time b) flight path seen from above during a commanded 7G level turn initiated at $Ma = 0.90$ **with a generic F-16 at a 5 percent lowered weight.**

time by 3 percent and reduced the loop radius by 4 percent, but the influence on load factors and velocities were negligible.

The only strong parameter in the maneuver was the maximum lift coefficient that is utilized for almost all the time, and the effects were clearly nonlinear. A 10 percent reduction in $C_{L_{max}}$ did not change the maneuver time, but increased the exit Mach number by 22 percent, maximum load factor by 19 percent and loop radius by 10 percent, as can be verified by comparing Figs. 3 and 8. A 10 percent increase in the parameter in turn lengthened the time to complete the maneuver by 5 percent while having a much smaller effect on the other output parameters than the reduction of the coefficient.

The variations in the roll performance had again small effects on the overall maneuver. A 30 percent change in the roll rate or time constant caused a 1 percent change in the maneuver time via the changes in the initial half roll.

Flat Scissors. In the flat scissors, weight variations did not affect the final velocity or the distance covered appreciably. Instead, a 5 percent weight reduction decreased the average turn radii by 4 percent, which means that the aircraft could complete more turns in the specified time interval. An increase in weight produced similar, but opposite effects.

A 10 percent drag reduction in turn increased the final Mach number by 8 percent, distance covered by 5 percent and the load factor by 17 percent at most. In contrast, the turn radii did not change. A drag increase brought about expected, reversed changes, and thrust variations had similar, but slightly smaller effects.

With the control strategy specified in this work, the available performance improvement from a weight reduction was used to tighten the turns at an essentially constant speed, whereas the improvements in drag or thrust were used to increase speed at constant turn radii. Although the control is not realistic for an actual air combat, hampering a comparison of W, T and D effects, it

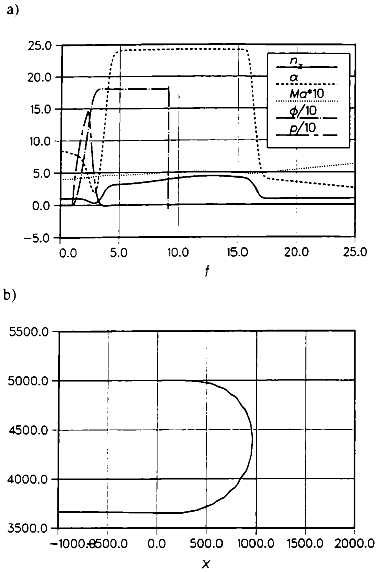


Fig. 8: a) Most important state variables as functions of time b) vertical projection of flight path during a split-S initiated at $Ma = 0.40$ with a generic F-16 having a 10 percent reduced $C_{L_{max}}$.

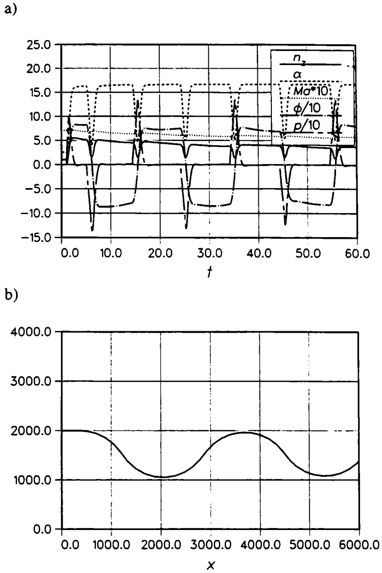


Fig. 9: a) Most important state variables as functions of time b) flight path seen from above during flat scissors initiated at $Ma = 0.70$ with a generic F-16 having a 10 percent reduced $C_{L_{max}}$.

can be concluded from the results that the weight variations are the most important of the three. The drag is only slightly less important, with the thrust following closely behind.

The maximum lift coefficient is the most important single variable, since the specified load factor is directly proportional to it. A 10 percent reduction in $C_{L_{max}}$ increased the final velocity by 20 percent, the distance covered by 16 percent and the average turn radii by 13 percent. At the initial stages of the maneuver, the load factor reduced by 10 percent but increased by 33 percent at the end. These large changes become evident via a comparison of Figs. 4 and 9. An increase of the maximum lift caused somewhat less dramatic effects.

As the flat scissors include several direction changes, the roll performance was expected to be a significant factor in the maneuver. However, the observed sensitivities were rather low. The largest effect was obtained by reducing the maximum roll rate by 30 percent, which led to an increase of 3 percent in the effective, average turn radii. The effects of the other variations, though, were almost of the same magnitude.

Rolling Scissors. In the series of barrel rolls emulating a rolling scissors maneuver, the time of each revolution and the load factor were fixed because of difficulties in the control definition. Thus, only the aircraft velocity was a relevant variable, and the Mach number after one minute was monitored.

Once again the aircraft weight was more important than thrust or drag. A weight increase of 5 percent reduced the final velocity by 8 percent. A corresponding weight reduction increased the velocity by only 2 percent, because the aircraft was accelerating at Mach numbers related to the transonic drag rise. In the acceleration, the sensitivity to the thrust was slightly higher than to the drag, both being about 2/3 of the sensitivity to the weight.

Defensive Spiral. In the spiral performed at idle, the idle thrust itself proved to be insignificant. The 10 percent thrust variations did not have a noticeable effect on the results. Instead, drag variations produced clear differences. The 10 percent drag reduction increased the final Mach number by 9 percent and the load factor by 17 percent at the end of the simulation after one minute. The total heading change increased by 12 percent and the altitude loss by 7 percent. A corresponding drag increase did not affect the final velocity and load factor appreciably, but reduced the heading change by 5 percent and altitude loss by 7 percent.

The effects of weight variations were relatively mild. A weight reduction increased the turn rate slightly, and a 5 percent increase slowed down the rotation by 5 percent and increased the altitude loss by less than 2 percent.

Clearly the most important variable was the maximum lift coefficient, the 10 percent reduction of which increased the final Mach number by 15 percent and the load factor by 26 percent, accelerated the rotation by 12 percent and increased the altitude loss by 14 percent. The effects of an increase in $C_{L_{max}}$ were of somewhat lesser magnitude.

Turn Initiation. In this case, only the roll performance that was of minor importance in the preceeding maneuvers was studied. A 30 percent increase in the maximum steady-state roll rate increased the achieved peak roll rate by 10 percent and shortened the time to achieve the desired bank angle by about 6 percent. A corresponding reduction of the roll limit reduced the peak roll rate by 18 percent and lengthened the maneuver time by the same percentage.

A 30 percent reduction of the time constant increased the peak roll rate by 20 percent and cut the time to achieve the final state by 12 percent. The increase in the time constant in turn reduced the peak roll rate by 13 percent and increased the total time by 24 percent.

A reduction in the roll performance is found to be more influential than an improvement. The effects are strongly nonlinear. Furthermore, at least in this case, the time constant is a slightly more important parameter than the steady-state roll rate. The practice of defining p_{max} as a three-dimensional array and τ_p as a single constant, as done in the AML codes, appears to be badly out of balance.

2.6 Relative Importance of Model Parameters

In this section, general conclusions concerning the relative importance of aircraft model parameters in different maneuvers are made.

Broadly speaking, the most important aircraft parameter seems to be its weight. Besides the turn initiation, only the spiral was a maneuver where the weight was not among the two most powerful variables. In all situations where the specific excess power is important, the weight has a decisive effect. This conclusion is significant for the definition of accuracy requirements for the other model parameters. If the weight is not expected to be known at a better than 5 percent accuracy, there is no point in requiring the other factors to be more accurate.

In maneuvers depending on the lift boundary, the maximum lift coefficient is at least as important as the weight. Such situations are encountered in rapid turns at speeds representative of an air combat. In the maneuvers studied, the maximum lift coefficient was the most important variable in the tight subsonic turn, split-S, flat scissors and spiral. However, owing to the limitations of the present approach, its total impact in actual air combats remains obscure.

The maximum thrust and the drag polar are important factors, but generally less critical than the quantities discussed above. Their mutual importance depends on whether the aircraft is accelerating or decelerating. In the level acceleration, the thrust was the most important variable, and the drag was dominant in the supersonic turn. The idle thrust appears to be a very weak parameter, easing up the modeling effort on its part.

Except the turn initiation, the roll performance was always much less important than the other variables considered. In the maneuvers studied, the expected inaccuracies in the maximum roll rate and the roll time constant are overwhelmed by the errors in the more powerful characteristics. It is difficult to imagine any combination of events lasting for the order of tens of seconds, where the roll performance would be essential in comparison with the other factors. The mutual importance of the steady-state roll rate and the time constant in turn seem to be roughly equivalent. It is not reasonable to model one of these in a detailed manner if the other is almost ignored.

3 Steady-State Performance Calculations

3.1 Problem Set-Up

The aircraft model studied in the performance calculations is the same generic F-16 as in the maneuver simulations.

The evaluations were made in the ISA conditions covering the whole flight envelope, defined by the Mach number and altitude. Three performance indicators are considered. The specific excess power $SEP = (T - D)V/W$ was calculated at the load factor of unity applying maximum afterburner. The maximum steady-state load factor n_{z_s} in a level turn was computed at the maximum thrust without taking the normal component of the thrust into account. This simplification causes

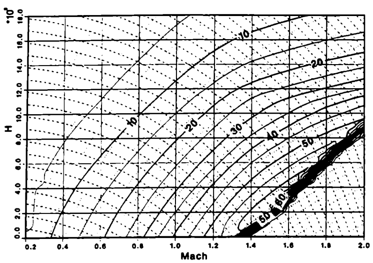


Fig. 10: Improvement in specific excess power [m/s] due to a 10 percent thrust increase with a generic F-16.

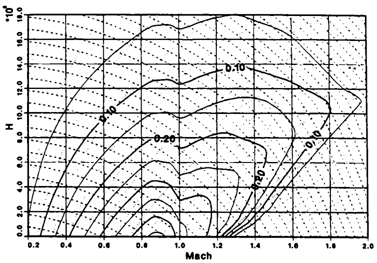


Fig. 11: Improvement in sustained load factor due to a 5 percent weight reduction with a generic F-16.

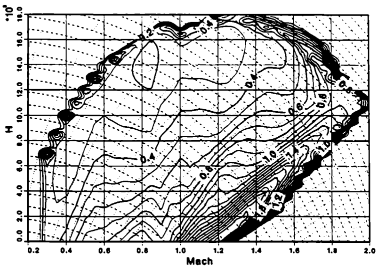


Fig. 12: Improvement in sustained turn rate [°/s] due to a 10 percent drag reduction with a generic F-16.

inaccuracies in the absolute results but does not affect the comparisons. The third performance quantity is the steady-state turn rate $\dot{\Psi}$, the calculation of which corresponds to the load factor.

In the sensitivity analysis, the variables were the aircraft weight, thrust and drag, which were modified similarly as in the maneuver simulations.

3.2 Sensitivity of Performance Indicators

The specific excess power is sensitive to the thrust variations. The effects become rapidly stronger with an increasing equivalent airspeed, as shown in Fig. 10. The effects of drag variations are small at low altitudes and airspeeds, but they increase quickly in the supersonic region, becoming as important as thrust variations near the right and upper boundary of the flight envelope.

The distribution of the effects of weight variations is markedly smoother in relation to the thrust or drag. The sensitivity is at its maximum at low altitudes at high subsonic speeds, but even then it is clearly weaker than the sensitivity to the thrust. At low equivalent airspeeds, the W and T sensitivities are of the same order, but in the supersonic region, weight becomes relatively unimportant. In relation to the drag effects, the weight effects are larger in the subsonic region and much smaller at supersonic speeds.

The maximum level-turn load factor n_{z_s} depends on the thrust and drag qualitatively similarly as SEP, but now their mutual importance is identical. The sensitivities increase rapidly with equivalent airspeed. The effect of the weight variations, shown in Fig. 11, are of the same shape as the contours of the load factor itself. In the subsonic region, the W sensitivity is about twice as high as the T and D sensitivity, but in the supersonic range, the latter two grow quickly much higher than the former.

With respect to the turn rate, the sensitivities behave similarly as in the n_{z_s} study. The thrust and drag effects are equivalent and increase generally with airspeed. The effects of drag variations are illustrated in Fig. 12. The sensitivity to the weight variations is highest at sea level and at subsonic speeds, where it is about double in relation to the thrust and drag variations. At supersonic speeds, instead, the weight is a weak parameter.

3.3 Relative Impact of Aircraft Characteristics

In the steady-state performance evaluations, the weight is not as dominant as it was in the maneuver simulations. This results partially from the absence of axial accelerations, and on the other hand, the (Ma, H) plots cover the whole flight envelope, emphasizing the supersonic region in comparison to the maneuver studies. The weight is still important, and in turning flight at subsonic speeds, it has the strongest effects of the variables considered.

The thrust is the most important parameter when the energy altitude is to be increased, especially at high speeds with high drag. In steady-state turns, the sensitivities to the thrust and drag are naturally similar, because in the basic equilibrium, the relation $T = D$ holds.

4 Conclusions

The accuracy requirements for performance models of fighters aimed at air combat simulations were studied by conducting maneuver simulations with the improved AML-75 code and by computing steady-state performance in the whole flight envelope, applying systematic variations of model parameters. From the sensitivity studies, several conclusions can be drawn.

In an air combat within visual range, the most important single model parameter is aircraft weight, which must be known with an accuracy of a few percent to obtain representative results. As the weight cannot be reliably estimated on the basis of an external geometry, other sources of information become critical. On the other hand, since the maneuverability in several conditions depends rather directly on weight, small weight adjustments may be used to compensate for errors in the other model parameters.

The maximum thrust and the drag polar are mutually important parameters, but in the subsonic region, aircraft weight is more critical than them. In the supersonic region, the thrust and the drag are dominant. They should be modeled within 5 percent of accuracy for reasonable results. In contrast, the idle thrust appears to be a very weak parameter, and not much effort should be spent on estimating it.

The maximum lift coefficient is the most important aircraft parameter in many air combat maneuvers performed below the corner speed. The modeling accuracy should be of the order of 5 percent to be able to predict well the motion at the lift boundary. This requirement appears to be tough to meet if the aircraft model is to be built just on the basis of the external shape. On the other hand, the overall importance of this parameter in real combat remains open in the light of this study.

The roll performance appears to be of minor importance compared to the parameters discussed above. Relatively large, 30 percent changes in the rolling capability are overshadowed by errors of a few percent in the weight or drag, even in relatively short-duration maneuvers. The maximum steady-state roll rate and the roll time constant are mutually important if the motion is modeled as described in this paper, implying that it is illogical to concentrate the modeling effort on just one of these, as is to be done with simulations using standard AML codes.

As envisaged, the findings of the study help to concentrate available effort on the most important aspects of a model. Overall, a practical threat modeling appears to be a difficult but probably manageable task.

5 Acknowledgement

This work was funded by the Finnish Air Force.

References

[1] Shaw, R., *Fighter Combat, Tactics and Maneuvering*. Naval Institute Press, Annapolis, MD, 1985.

[2] Burgin, G., Fogel, L., and Phelps, J., "An Adaptive Maneuvering Logic Computer Program for the Simulation of One-on-One Air-toAir Combat, Vol. I: General Description," NASA CR-2582, Washington, DC, 1975.

[3] Burgin, G. and Owens, A., "An Adaptive Maneuvering Logic Computer Program for the Simulation of One-on-One Air-toAir Combat, Vol. II: Program Description," NASA CR-2583, Washington, DC, 1975.

[4] Kahaner, D., Moler, C., and Nash, S., *Numerical Methods and Software*. Prentice Hall, 1988.

[5] "Jane's All the World's Aircraft." 1991-92.

[6] Kurzke, J., "DASTURB version 6.0, A Program to Calculate Design and Off-Design Performance of Gas Turbines," Dachau, Germany, 1995.

[7] "USAF Stability and Control DATCOM." Revision 1978.

[8] Raymer, D., *Aircraft Design: A Conceptual Approach*. AIAA, Washington, DC, 1992.

[9] Hoffren, J. and Vilenius, J., "Calibration of Air Combat Simulation Models Based on Performance Data." in *Proceedings of ICAS98*, (Melbourne, Australia), ICAS-98-1.11.2, September 1998.

AIAA-98-4166

SIMULATION OF AIRCRAFT FLIGHT DYNAMICS BY MEANS OF DYNAMICALLY SIMILAR MODELS.

S. Sadovnychiy, Ph.D., National Politechnical Institute of Mexico. Mexico, D.F.,
A. Betin, Ph.D., Scientific & Research Institute for Problems of Physical Modeling, Kharkov, Ukraine.
A. Ryshenko, Dr. Tech. Science, Kharkov Aviation Institute, Kharkov, Ukraine.
Ricardo Peralta-Fabi, Ph.D., National University of Mexico. Mexico, D.F.,

ABSTRACT

This paper deals with methods for physical modeling of aircraft flight by means of Dynamically Similar free-flying Models (DSM) useful for the investigation of aircraft flight dynamics with structural and/or onboard system failures. The technique for the development of conceptual design is offered. Four variants of compliance with similarity criteria of Froude, Reynolds and Mach numbers are considered. The method for the definition of actual working parameters has been developed. As an example, the diagrams of flight parameters of the SU-7 aircraft model, with 40% in-flight loss of wing span, are presented.

Keywords:

simulation, modeling, aircraft, scale coefficients, criteria.

INTRODUCTION

The experimental investigation of an aircraft's flight dynamics with structural and/or onboard system failures is an indispensable component of an emergency situation analysis. Wind tunnel tests, as well as other traditional methods of experimental aerodynamics can hardly satisfy an adequate modeling of the complex spatial dynamics of modern aircraft under emergency in-flight situations, its violent maneuvers and and flight at angles of attack beyond stall. The implementation of this kind of tests on real aircraft is expensive and carries a risk to the test crew.

On the other hand, flight simulation by means of large scaled free-flying, dynamically similar models (DSM) provides access to the most complete data about diverse emergency flight situations in a short time [1,2,3,4].

STAGES OF DEVELOPMENT

In particular, DSM development depends on the need for a correct, scientifically proved, settlement of compromises between the requirements of an adequate modeling of the phenomenon under investigation, and its technical implementation. The theoretical basis for DMS development depends on complying with a system of necessary, sufficient, complete and agreed similarity criteria. The definition of this system is the main task of a conceptual design stage. On subsequent stages the similarity demands complicate the design process by further introducing a system of compromising relationships into the parameters of design.

The proposed technology of conceptual design allows for the solution of these contradictions within the technical feasibility and limitations of the DSM. For the determination of rational values of the DSM the goal is to ensure equivalence between possible and necessary parameters. In other terms, a set of necessary test flight characteristics of the DSM should be fulfilled and the possible aircraft flight envelope for the simulation should also be provided. The final selection of the main working parameters from the group of vectors within rational values is carried out by means of an optimization procedure that depends on a chosen goal function.

CONDITION OF SIMILARITY

In the proposed method the main parameters of a DSM are the following:

L - characteristic linear dimensions,
m - take-off mass,
J$_i$ - moments of inertia.

For a correct representation of the processes under investigation the DSM should fulfill similarity conditions. Thus, the parameters of a given DSM should provide not only geometric and kinematic, but also the dynamic similarity, according to the criteria that defines the phenomena. Usually during the aircraft flight dynamics simulation, besides Newton and Strukhal criteria, the possibility of satisfying the Froude, Reynolds and Mach criteria is also under consideration[5,6]. For satisfying the Newton dynamic similarity criterion, the scales of principal parameters of DSM should be:

$$\begin{aligned} \mathbf{K_m} &= \mathbf{K_\rho \bullet K_L^3} \\ \mathbf{K_J} &= \mathbf{K_\rho \bullet K_L^5} \end{aligned} \qquad (1)$$

where **K** is a scale coefficient (i = L, ρ, m, J), and ρ is mass density.

There are four variants for satisfying the Froude (**Fr**), Reynolds (**Re**) and Mach (**M**) criteria.

1. Satisfying simultaneously all principal criteria of the disturbed flows (**Fr +Re+ M**). It is possible only when the altitudes (**H**) of the real aircraft (R) and DSM (m) are equal, $\mathbf{H_R = H_m}$ with linear dimensions scale $\mathbf{K_L}$ = 1 . Consequently $\mathbf{K_m}$ = $\mathbf{K_J}$ =1. It does not mean that a DSM should be built as the real aircraft. The DSM should have the same geometry, mass and moments of inertia, but it could be made of different materials, scale and have special equipment and systems for tests.

2. Satisfying only Froude criterion. It is acceptable only for studies of flight regimes with parameters which are inside the flight envelope of autosimilarity on **Re** and **M**. This is a unique case where the choice of linear dimensions, scales and test altitudes are independent. But afterwards we should fulfill relationship (1).

   Let us suppose that the group of $\mathbf{K_\rho}$ scale values satisfies the **Fr** criterion. Therefore, for any of these values the increment of scale $\mathbf{K_L}$ produces an increase of the values of scales $\mathbf{K_m}$ and $\mathbf{K_J}$, or the growth of the necessary values of mass and axial moments of inertia for the DSM. The minimum required values exist for both $\mathbf{K_L = K_{L\ min}}$ and $\mathbf{K_\rho = K_{\rho min}}$. If some restrictions were applied to the mass and the determination of two iteration cycles were necessary: the variation of $\mathbf{K_L}$ from $\mathbf{K_{L\ min}}$ to $\mathbf{K_{L\ max}}$ (outer cycle) and the variation of K_ρ from $\mathbf{K_{\rho\ min}}$ to $\mathbf{K_{\rho\ max}}$ (inner cycle), then the first value of mass $\mathbf{m_m}$ which satisfies the restrictions will be minimum and necessary. Also the axial moments of inertia and scale $\mathbf{K_L}$ will be minimum.

3. Satisfying the Froude and Reynolds criteria for the implementation of flight tests with parameters which are inside the envelope of autosimilarity on **M**. In this case:

$$\begin{aligned} K_L &= \sqrt[3]{\frac{\nu_m^2}{g_m} \cdot \frac{g_R}{\nu_R^2}}\,; \\ K_m &= \frac{\nu_m^2 \rho_m}{g_m} \cdot \frac{g_R}{\nu_R^2 \rho_R}\,; \\ K_J &= \left(\frac{\nu_m^2}{g_m} \cdot \frac{g_R}{\nu_R^2} \right)^{\frac{5}{3}} \cdot \frac{\rho_m}{\rho_R}, \end{aligned} \qquad (2)$$

where ν_i are coefficients of the kinematic viscosity of air at flight altitudes of the DSM (index m) and the real aircraft (index R): g_i are the g values at the same altitudes.

4. Satisfying the Froude and Mach criteria. In this case:

$$\begin{aligned} K_L &= \frac{T_m}{T_R} \\ K_m &= \frac{T_m^3 \rho_m}{T_R^3 \rho_R} \end{aligned} \qquad (3)$$

$$\mathrm{K}_J = \frac{T_m^5 \rho_m}{T_R^5 \rho_R}$$

In all cases of satisfying the criteria in pairs (**Fr** and **Re** or **Fr** and **M**) with arbitrarily selected values of $\mathbf{H_R}$, the increment of $\mathbf{K_L}$ leads to the growth of $\mathbf{K_m}$ and $\mathbf{K_J}$. The increment of $\mathbf{K_L}$ (with $\mathbf{H_R}$=*const*) introduces the need to fulfill the tests of the DSM at a higher altitude $\mathbf{H_m}$. All these techniques under consideration allow for the simulation in the envelope of autosimilarity of one of the criteria, the regime V=*const*, H=*const*, as well as maneuvers with V=*var* and H=*const*. Thus, during the conceptual design of the DSM, to achieve the minimum of the goal function $\mathbf{m_m}$ it is necessary to arrive at a DSM with linear sizes as small as possible.

By setting various values of the test altitude it is possible to determine the similarity scales of DSM according to the accepted criteria.

TECHNICAL FEASIBILITY

The determination of technically feasible parameters can be divided into three stages. The first is the determination of minimum mass. At an early of stage design, when the loading on the structure is unknown, minimum mass (in a first approximation) can be calculated by the formula:

$$M_m{}^I = \frac{M_e + M_{eq}}{1 - \bar{M}_s - \bar{M}_p - \bar{M}_f - \bar{M}_{c.g.}} \quad (4)$$

where : $\bar{M}_s$ is the equivalent mass of a structural model, M_e is the engine mass , M_{eq} is the equipment mass, $\bar{M}_p$ is the equivalent mass of the parachute system, $\bar{M}_f$ is the equivalent mass of fuel, and $\bar{M}_{cg}$ is the equivalent mass of loads placed for the correction of the centre of graviaty.

All parts of this equation can be determined on the basis of statistical data and formulas from /7/. The exact calculation of minimum mass $M_m{}^{II}$ (in the second approximation) takes into account the dependence of structural mass on size, peculiarities of configuration, and loading conditions. It is not possible to solve equation (4) relative to $M_m{}^I$ in a direct (final) form, thus it is necessary to apply a method of iterations. The determination $M_m{}^{II}$ can be presented as:

$$M_m{}^{II} = \frac{M_e + M_{eq} + M_p + M_f}{1 - \bar{M}_s - \bar{M}_{c.g.}} \quad (5)$$

and the equivalent mass of a structure is determined by means of a formula offered by Caddell W.E. /8/.

$$\bar{M}_s = A\sqrt{n_z^c}\left(\frac{B_f H_f L_f}{M_m}\right)^{0.24} \quad (6)$$

where A is a coefficient dependent on the aerodynamic scheme of the real vehicle and the presence of the engine, $n^c{}_z$ is the computational overload on the Z axis, and B_f, H_f, L_f are the width, height and length of the DSM. The obtained minimum mass of a DSM is taken as the theoretically minimum mass for further calculations.

The second stage is the determination of linear dimensions of the DSM or their scale relationships. The external contours of DSM are geometrically similar to contours of the real vehicle. The internal volume of a DSM should provide place for the structural elements, equipment, engine, etc. The linear scale of the model must comply with the volume needed for the installation of internal equipment, and can be determined by means of this formula:

$$K_{LW} = \sqrt[3]{\frac{W_{eq} + W_e + W_p}{W_{\Sigma m}\xi_m}} \quad (7)$$

where: M_i is the volume of equipment, engine, and parachute, $W_{\Sigma m}$ is the internal volume of the real vehicle, ξ_m is the statistical coefficient of density distribution. This coefficient is equal to the relation of a sum of volumes of the onboard equipment, engine, and structure to the internal volume .

But, on the other hand, it must provide an opportunity for the placement of all equipment for a desired centre of gravity. The "local" scales of each element of the equipment, takes into account a reserve factor for the structure, and is determined by the following formula:

$$K_{Li} = \frac{L_{mi}}{L_{Ri}} = \frac{(1.05\ldots1.15)L_{mi}}{L_{Ri}} \quad (i=1,2,3\ldots) \qquad (8)$$

The largest numerical dimension of K_{Li} determines this group of parameters. Thus, we have two scales: a scale based on a condition of compliance with internal volume, and a scale dependent on a desired center of gravity. The scale of a DSM, K_L should not be less than the largest of these two scales. Scale $K_{Lmin.}$ determines the minimum, technically feasible, size of a DSM.

$$L_{Mmin.} = L_R K_{Lmin.}$$

The third stage of analysis of technical feasibility, is the determination of the moments of inertia of the model. They can be determined by means of the well known formulas:

$$I_{xm} = \frac{M_m L_{Wm}^{\ 2}}{12}\varphi_{xm}; \qquad (9)$$

$$I_{zm} = \frac{M_m}{12}(L_{Wm}^{\ 2} + L_{Fm}^{\ 2})\varphi_{xm};$$

$$I_{Ym} = \frac{M_m L_{Fm}^{\ 2}}{12}\varphi_{xm}$$

where, L_{Wm} is the length of the wing of the model, L_{Fm} is the length of the fuselage, and φ_{xm} is a statistical coefficient.

The vector of the basic parameters

$$Q_{Ti}^{\ b} = Q_{Ti}^{\ b}(L_{m.}^{\ b} M_m^{\ b}, I_{Xm}^{\ b}, I_{Ym}^{\ b}, I_{Zm}^{\ b})$$

can differ from technically feasible parameters

$$Q_{Ti} = Q_{Ti}(L_{m.} M_m, I_{Xm}, I_{Ym}, I_{Zm})$$

but its parameters should satisfy the accepted similarity criteria and the conditions of technical feasibility

$$L_m \ge L_{m\,min}\, or K_L \le K_{Lmax};$$

$$M_m = M_m(K_L); I_{jm} = I_{jm}(K_L), j = X, Y, Z.$$

The correction of the axial moment of inertia by means of additional loads $M_{c.g.}$ should also be technically feasible, and their placement should not adversely affect the centre of gravity.

By carrying out a calculation for a number of optional K_L and K_r, it is possible to produce a group of vectors

$$Q_{Ti} = Q_{Ti}(L_{m.} M_m, I_{Xm}, I_{Ym}, I_{Zm})$$

that satisfies the accepted similarity criteria and the conditions of technical feasibility. In the following design stage, from a given group of vectors Q_{Ti}, we can find a Q_T, that satisfies flight-technical parameters (conditions of manufacturing, take-off conditions, maintenance service, preparation for flight) and flight limitations (speed, overload, height).

After that, for the justification of technical feasibility of a DSM the main necessary parameters are calculated by:

$$\mathbf{L_m = L_R K_L;\quad m_m = m_R K_m;} \qquad (10)$$

$$\mathbf{J_{im} = J_{iR};} \quad (i = x, y, z)$$

Then the flight parameter limitations are defined and, finally, the work vector of the principal parameters is obtained.

EXAMPLE

As an example of the proposed technique the flight tests of a DSM of an SU-7 aircraft (classic aerodynamic scheme) with in-flight loss of 20%, 28%, 40%, 50% of the wing span, permitted the determination of the critical damage for various flight regimes with the presence of damage to one side of the wing.

V= 65 m/s, H = 550m.

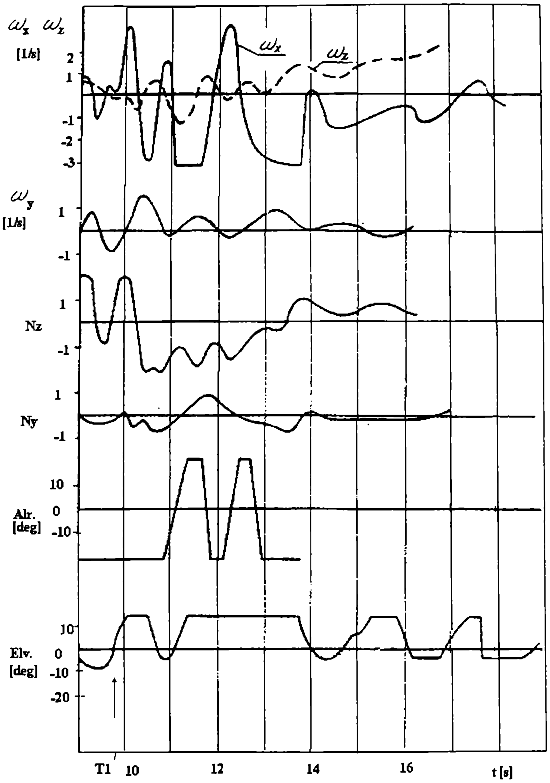


Fig.1. Parameters of flight of DSM

Fig.1 shows the change of flight parameters after the beginning of damage simulation (T1) of a wing (40 % of one half of the wing span). In this Fig. Elv. is the deflection of the elevator; Alr. the deflection of the aileron; Ny, Nz the lateral and normal overload; ωx, ωy, ωz the angular velocity; and V the actual velocity of the model.

With a 28% loss, the flight control system, using other undamaged controls, managed to compensate for short term disturbances of the DSM, overshoot of the angular velocity about the X body axis was no more than 0.35 1/sec with a period of 0.4 sec. After this disturbance the flight of the DSM showed a relatively stable behavior.

References

1. Holleman E.C. Summary of flight tests to determinate the spin and controllability characteristics of a remotely piloted, large-scale (3/8) fighter airplane model. - NASA TN D-8052, 1976. 124 p.
2. Flight tests with piloted models slated. - Aviation Week & Space Technology, 2.10.72, p. 55-57.
3. Michael A. Dornheim. X-36 To Test Agility of Tailless Design. - Aviation Week & Space Technology, March 4, 1996, p. 20-21.
4. Patuxen River. NASA Drop Model Sinks. - Aviation Week & Space Technology, January 20, 1997. P.59.
5. Sedov L.I. Similarity methods and dimensions in mechanics. - M., Nayka, 1981.
6. Stephen J. Kline, Similitude and Approximation Theory.- McGraw-Hill Book Company, N.Y.
7. Betin A, Rushenko A., Ryabkov V., Cheranovski O. Determination of the sizes, mass and inertial parameters of DSM. The educational text-book for students. Kharkov Aviation Institute, Kharkov, 1992.
8. Caddell W.E. On the use of aircraft density in preliminary design. SAWE, Paper No. 813, May 1969.

AIAA-98-4167

SIMULATION MIDDLEWARE OBJECT CLASSES (SMOC) FOR SIMULATION & MODELING APPLICATIONS

Daniel J. Paterson
Naval Air Warfare Center
Training Systems Division
Orlando, FL 32826-3224
407-380-8564
dan_paterson@ntsc.navy.mil

ABSTRACT: Naval Air Warfare Center Training Systems Division has produced a product to facilitate the development and integration of simulation/modeling applications to interoperate seamlessly in an environment of heterogeneous communication protocols. The product is a set of Simulation Middleware Object Classes (SMOC) that provide an interface between the simulation application and the simulation interoperability protocol, which may include Distributed Interactive Simulation (DIS IEEE 1278.1), various Federation Object Model (FOM) implementations of the High Level Architecture (HLA), and other protocols as necessary. The SMOC can operate either as a stand alone gateway, or as an integrated Middleware that is embedded in the simulation application. The paper will focus on the SMOC architecture and testing issues.

INTRODUCTION

The project developed a Simulation Middleware Object Classes (SMOC) architecture which can be configured as a Gateway or used as Middleware in an application. The SMOC provides the conversion from DIS to HLA, or from any simulation interoperability protocol in the Gateway configuration. In the Middleware configuration, the application talks directly to the SMOC API which performs the HLA, DIS or any simulation interoperability protocol functions. This effort was an internal NAWCTSD project sponsored by the Office Of Naval Research (ONR).

HIGH LEVEL ARCHITECTURE

The shift from a DIS paradigm was prompted by the recently developed DoD High Level Architecture (HLA). The HLA Baseline Definition was completed on 21 August 1996 as a DoD initiative. It was subsequently approved by the Under Secretary of Defense for Acquisition and Technology [USD (A&T)] as the standard technical architecture for all DoD simulations, on 10 September 1996. [1]

The project converted a DIS-compliant simulator into an HLA-compliant simulator. The simulator we converted is a multi-use generic "F-14" simulator which we have used in DIS exercises as an F-14, an F-18, and a Mig-29. Our approach was to establish a federation comprised of multiple aspects of this same simulation operating together through our laboratory LAN. Federates were instantiated so that we could demonstrate and test HLA during air-to-air combat maneuvers.

HLA Rules of Usage

There are a total of ten HLA rules, five for federations and five for federates. [1]

Rules for Federations are:

Rule #1: Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the HLA Object Model Template (OMT).

Rule #2: In a federation, all object representation shall be in the federates, not in the runtime infrastructure (RTI).

Rule #3: During a federation execution, all FOM data interchanged among federates occurs via the RTI.

Rule #4: During a federation execution, federates shall interact with the RTI in accordance with the HLA interface specification.

Rule #5: During a federation execution, an attribute of an instance of an object shall be owned by only one federate at any given time.

Rules for Federates are:

Rule #6: The federate shall have an HLA Simulation Object Model (SOM), documented in accordance with the HLA Object Model Template (OMT).

Rule #7: Federates shall be able to update and/or reflect any attributes of objects in their SOM and send and/or receive SOM object interactions externally, as specified in their SOM.

Rule #8: Federates shall be able to transfer and/or accept ownership of attributes dynamically during a federation execution, as specified in their SOM.

Rule #9: Federates shall be able to vary the conditions (e.g., threshold) under which they provide updates of attributes of objects, as specified in their SOM.

Rule #10: Federates shall be able to manage local time in a way which will allow them to coordinate data exchange with other members of a federation.

HLA Object Model Template

The next most significant aspect of HLA compliance revolves around object models. We will build HLA object models which conform to the HLA Object Model Template (OMT). [2]

We did use the Object Model Development Tools (OMDT) (Beta version), [3] built by AEgis which were developed for use with the Joint Training protofederation. We used these tools to document our FOM and develop our SOM. We recognized early in the process that it may be necessary to tailor the scope of the Realtime Platform Reference Federation Object Model (RPR-FOM) [4] to meet our research needs. In fact, our FOM is a very specialized subset of the RPR-FOM optimized for our F-14 trainer needs.

Previously, it was mentioned that we did use the RPR-FOM for our investigation. This is driven by two concerns; a desire to reuse DIS derived data and a need to expedite our SOM development process.

We feel that the RPR-FOM represents a best practice product that uses a broad set of indices which describe common military operations.

Simulation Middleware Object Classes (SMOC) Architecture

The SMOC Architecture focused on capitalizing on the benefits of an object oriented design approach, and attempted to accommodate a degree of reusability, flexibility, and scalability.

The basic idea was to create a set of object classes that could be configured as necessary to act either as a stand alone gateway, or as a layer of Middleware for a simulation application to provide a seamless integration across DIS and the potential variations of HLA FOM environments. The classes were designed to be reusable, adaptable and scaleable. Figure 1 shows a simplified representation of the implementation using the Middleware object classes.

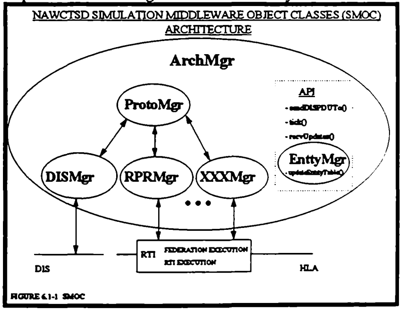


FIGURE 1 SIMULATION MIDDLEWARE OBJECT CLASSES

With the new simulation object classes, the user interface consists of an architecture configuration data file and an interface that provides the functions such as updateEntityTable(), tick(), sendDISPDU() and recvUpdates().

The architecture configuration data file is set up by the application designer, and contains definitions for each of the communications channels that the application will need. The general format of each definition is as follows:

```
federate_name federation_name federation_type data_transport_type [parameters]
```

As an example, assume that an application needs to participate in a DIS exercise. We will name the entity NASNET_F14, and the exercise DOGFIGHT_1. Also, we will assume that we will be connecting to this exercise via ethernet. In this case, the implementor would include a line similar to the following in the architecture data file:

```
NASNET_F14 DOGFIGHT_1 DIS ETHERNET 192.44.253.255 192.44.253.255 6994 6994 ....
```

Testing of the SMOC implementation

Test Objectives

The primary objective of our test program was to perform a representative flight test of a flight simulator against a second flight simulator and measure the performance of the SMOC in a Gateway mode. In addition a series of stress tests were designed to gradually increase the number of interactions at the DIS and HLA interfaces to again test the performance of the SMOC in Gateway mode.

Test Software and Hardware

In order to test the SMOC, NAWCTSD utilized several pieces of flight simulator code. We have a flight simulation software package modeling an F-14/F-18 aircraft which was written in FORTRAN and a display simulation software package for the F-14/F-18 aircraft cockpit which was written in C. The F-14/F-18 model has the ability to fire only one missile at a time, but allowed unlimited stores. Thus, it was a perfect testbed to engage in air-to-air "dogfights" over a network.

Since the SMOC was being developed under Microsoft Visual C++, the F-14/F-18 FORTRAN code had to be translated into C++. This was done by using the f2c utility (http://www.netlib.org/f2c/) that is available on the Internet in the public domain. The resulting translated code is not the prettiest C code, but neither was the original FORTRAN code that we were working with. The f2c program worked well, and with no problems, even though the original FORTRAN code took advantage of many FORTRAN "tricks" such as multiple EQUIVALENCE statements. The only additional step necessary to translate to C++ from the C code, generated by f2c, was to add function prototypes to all of the functions defined in the F-14/F-18 code. Since the flight model did not have any text input or output, it wasn't even necessary to link in the f2c libraries.

Once the flight model was successfully translated into C++, the code had to interface with the SMOC. Since the model already had a DIS interface with the NIU, it was easy to find all of the NIU API calls and translate them into SMOC API calls. In fact, all of the F-14/F-18's calls to the NIU were located in one file. Since all of the NIU API calls had direct SMOC equivalents (such as sending and receiving PDUs), it was an easy matter to make the F-14/F-18 SMOC compliant.

A second set of code is the F-14/F-18 aircraft cockpit display software simulation, written in C. The software simulation displays all the instrumentation's, radar and other various information necessary for the pilot to flight the aircraft. The cockpit display was running under the Linux Operating System using the X-Windows programming interface. In order to run the cockpit display under Windows NT, we decided to convert the F-14/F-18 cockpit display software from X-Windows to the Win32 Application Program Interface (API)

With the F-14 up and running, we now had a man in the loop simulator that was capable as Flying in a DIS or HLA exercise "natively." In addition, all pieces of the simulation now ran under a single operating system (Windows NT).

While the process described above is only one example, and while it was a relatively painless undertaking since the original application was integrated in the laboratory environment for research purposes, (and already using the NAWCTSD DIS Tool Set,) it does illustrate some of the issues involved in making a legacy system HLA compliant.

To complete the testing setup, a program called "Fly" was utilized. Fly is a generic Semi-Automated Forces (SAF) program that allows for the generation of HLA objects or DIS entities when interfaced with a copy of the SMOC set up in Middleware mode. The operator, through the use of a script file, can change the number of objects in the scenario and set up Fly to produce either HLA objects or DIS PDU's. The Fly program can generate multiple moving targets which are utilized to stress the SMOC's HLA or DIS interface.

All tests were done using the "debug mode" of Microsoft Visual C++ Programmer's Workbench and actual results should show less latency than indicated in the test result tables. In addition, the HLA - DIS test required a setup of the Fly program to force HLA interactions at constant rates. For example if the chart shows 30 HLA attribute updates, then the Fly program was producing a constant rate of 30 HLA attribute updates per second to be broadcast via the HLA RTI to the SMOC HLA interface. This type of test setup creates a stress test on the HLA interface to the SMOC, which may not naturally occur in a HLA environment.

Internal SMOC latency measurements were taken by operating the SMOC as a stand alone application in GATEWAY mode. In the case of translating DIS to HLA, latency was measured as the elapsed time between receiving a call from the UDP socket and returning from a call to the *updateAttributeValues()* or *sendInteraction()* methods of the RTI ambassador. In the case of translating HLA to DIS, latency was measured as the elapsed time between receiving a federate ambassador *reflectAttributeValue()*, or *receiveInteraction()* call from the RTI, and transmitting a DIS PDU on the UDP socket.

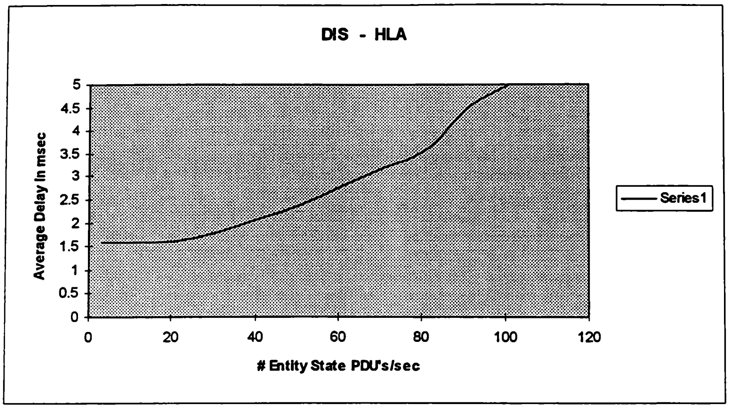
The test platform utilized Windows NT 4.0 and a 200 MHz MMX CPU. The SMOC was run in debug mode through the Microsoft C++ 5.0 Programmer's Workbench application. The platform was configured with two 10 MBPS ethernet cards, one for sending/receiving DIS data, and one for sending/receiving HLA data. The network cables for these two networks were physically connected in order to view network data with an ethernet packet "sniffer". As a result, both DIS and HLA data were transmitted on the same physical medium. The rtiexec and fedexec were both run on the same computer, a 200 MHz Pentium running Windows NT. Since the application was running on a PC, byte swapping also had to be performed on all data. It should also be noted that no effort was made to optimize the code for performance in this first version of the SMOC, and no scaling capability was utilized. Also, network and platform configurations could have been changed to increase performance.

Host	Platform	CPU	OS
DIS-Viewer	SGI - O2	R10000	IRIX 6.2
DIS	PC	Intel 200 MMX	NT
Gateway	PC	Intel 233 MMX	NT
HLA	PC	Intel 200 MMX	NT

Table 1 Test Hardware Description

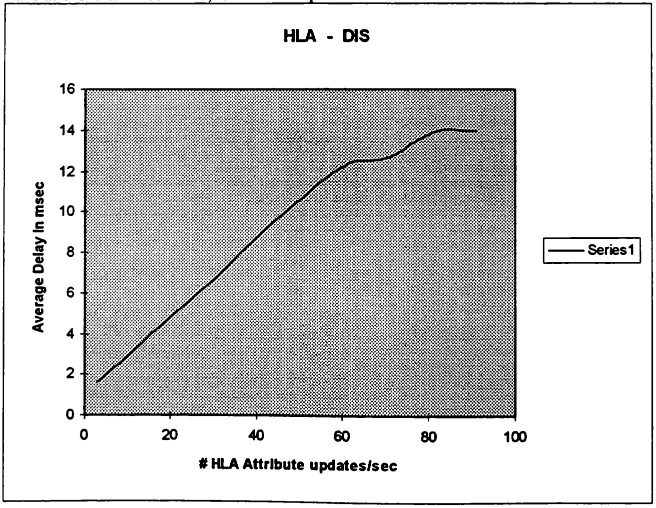
Performance results

Performance results shown below are from a set of preliminary tests run on the SMOC at NAWCTSD. These initial tests were accomplished as part of a test program set up to investigate the latency of the SMOC in Gateway mode. Latency is dependent upon the network and platform configuration among other things, as well as the number of entities/federate objects being processed.



SMOC Performance, DIS - HLA

With the setup described above, preliminary test results indicate that the SMOC gateway imposes a delay of less than 3 ms for small numbers of entities and objects. The latency was greater for HLA to DIS translations than vice versa, and went up as the number of entities and objects increased. At 100 entities/objects, HLA to DIS latency went up to around 14 ms, while DIS to HLA latency only went up to around 5 ms.



SMOC Performance - HLA to DIS

Additional testing of the SMOC involved a standard 1-on-1 dog-fight between HLA and DIS configured simulations. Internal latency measurements were collected for this test and again performance was not impacted. The latency for the 1-on-1 dog-fight through the gateway was less than 5 ms.

Testing of the SMOC in a BFTT configuration allowed us to test for voice throughput through the system. This set-up allowed for near-real-time delivery of audio through the HLA with no perceivable delays.

Testing of the SMOC in a video configuration allowed us to test for video throughput through the system. This system allowed for near-real-time delivery of video through the HLA with no perceivable delays.

Conclusion

Fielded legacy trainers, especially those which are not DIS compliant, introduce a wide variety of issues that have to be resolved in order to make them HLA compliant. Such things as proprietary hardware and software, outdated computer platforms and programming languages, make updating legacy trainers a technical challenge. There are difficulties in physically connecting geographically separated trainer systems, and differences in simulation fidelities between trainer systems. New requirements for additional capabilities that had not yet been resolved satisfactorily under DIS (such as data link, common scenario, common detection, etc.) and so on, which impact how distributed training will be performed are currently being investigated by the simulation community.

Notwithstanding the application and site specific issues, the primary concern in achieving interoperability is currently the need for simulation applications to speak the same language at some level. With the current state of the art in simulation technology, that means HLA. The SMOC is the core component in NAWCTSD's approach to achieving this goal. Where feasible, the SMOC can be integrated into applications to eliminate the need for a gateway. For legacy systems where design modifications are not feasible, the SMOC can be utilized in GATEWAY mode to implement a non-intrusive solution, as long as the added gateway latency is acceptable.

ACKNOWLEDGMENTS

The author would like to acknowledge NAWCTSD researcher's Mr. Eric Anschuetz and Mr. Mark Biddle for their technical contributions and development of the Simulation Middleware Object Classes.

REFERENCES

[1] HLA Rules, DMSO, version 1.0, 15 August 1996

[2] HLA Object Model Template, DMSO, version 1.0, 15 August 1996

[3] URL HLA OMDT : http://hla.dsmo.mil

[4] URL RPR-FOM: http://siso.sc.ist.ucf.edu/docref/rpr-fom/index.htm

Paper Number: **AIAA-98-4168**

THE ARMY EXPERIMENT IV (AE4) SIMULATION/C4I EXPERIMENTS

Mr. Joseph M. Brennan/Mrs. Rita Simons
Systems Engineers
U.S. Army Simulation, Training, and Instrumentation Command (STRICOM)
Orlando, Florida

Abstract

This paper provides a discussion on the challenges associated with linking simulations and C4I systems. This discussion is based on the results of the Army Experiment IV (AE4) experiments which were executed 4 August through 5 September 1997. During the AE4 experiments, interfaces between various simulations and Army Battle Command Systems (ABCSs) were evaluated. Three separate experiments were performed to study various challenges for using simulations to stimulate fielded Command, Control, Communications, Computers, and Intelligence (C4I) systems: A Corps Battle Simulation (CBS) experiment, a Corps Level Computer Generated Forces (CLCGF) experiment, and a Dismounted Battlespace Battle Lab (DBBL) experiment. These experiments overlapped sufficiently to provide "two way" information flow from division down to individual soldier level. Discussion of several interfaces between simulations and ABCSs is provided within the paper. Points of discussion include: the benefits of perceived data (over ground truth data) for supporting C4I training applications, and a discussion of how perceived data can be obtained; the benefits for utilizing a C4I/simulation based laboratory for providing controlled test environments; and the benefits of designing fielded C4I systems with training in mind. In addition technical challenges that require resolution is discussed as well as potential opportunities for "follow-on" work.

Overview

U.S. Army Simulation, Training, and Instrumentation Command's (STRICOM's) Advanced Distributed Simulation Technology (ADST) II contract was utilized to support Army Experiment 4 (AE4) coordination, integration, and some development activities.

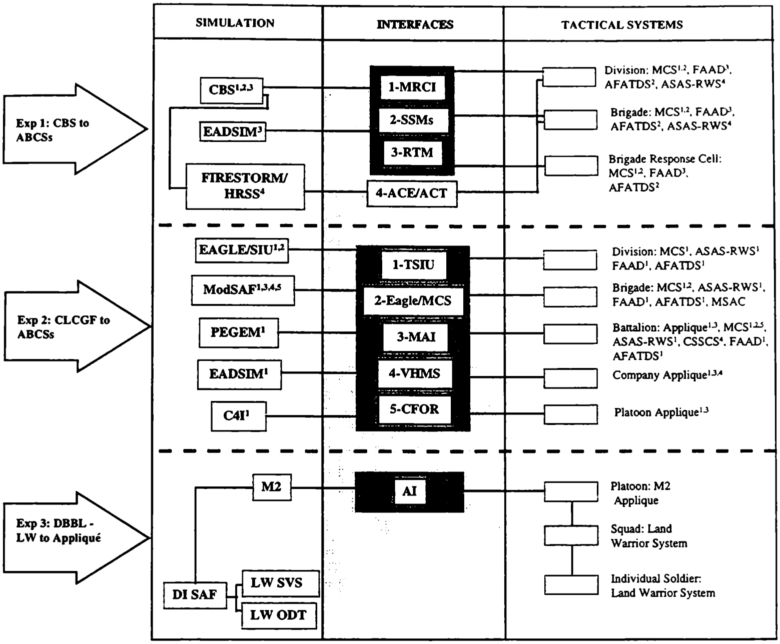
The ADST II contract tasks were performed by Lockheed Martin Information Systems and their subcontractors. The AE4 sponsor was United States Army Training and Doctrine Command (TRADOC). Several other organizations participated in the AE4 experiments. A complete list of participating Government and Contractor organizations is provided within the references cited at the end of this paper[1,2,3,4].

From 4 August through 5 September 1997, the AE4 team conducted a suite of experiments designed to create a working simulation capable of addressing some of the current challenges associated with linking simulations to live Army XXI tactical command and control (C2) systems from division level down to the platform/individual soldier levels. These experiments were broadly characterized as follows:

- Corps Battlefield Simulation (CBS) Experiment
- Corps Level Computer Generated Forces (CLCGF) Experiment
- Dismounted Battlespace Battle Lab (DBBL) Experiment

The AE4 experiment architecture is illustrated in Figure 2 below, and was comprised of simulation systems, interface technologies, and Army Battle Command System (ABCS) components.

The AE4 experiments provided a baseline simulation and tactical interface environment which provided valid and realistic insight into the linking of simulations with Army XXI C2 systems. These experiments were performed using a distributed simulation network (comprised of various simulations) and a separate tactical communications network (comprised of the Army Battle Command System (ABCS) version 2.0.4 (with patches) configured for each level of command). The simulation and tactical communications networks were interconnected utilizing various interfaces. These interfaces were evaluated for their potential for Warfighter training applications.


SIMULATION
INTERFACES
TACTICAL SYSTEMS
Exp 1: CBS to ABCSs
CBS^{1,2,3}
EADSIM^{3}
FIRESTORM/ HRSS^{4}
1-MRCI
2-SSMs
3-RTM
4-ACE/ACT
Division: MCS^{1,2}, FAAD^{3}, AFATDS^{2}, ASAS-RWS^{4}
Brigade: MCS^{1,2}, FAAD^{3}, AFATDS^{2}, ASAS-RWS^{4}
Brigade Response Cell: MCS^{1,2}, FAAD^{3}, AFATDS^{2}
Exp 2: CLCGF to ABCSs
EAGLE/SIU^{1,2}
ModSAF^{1,3,4,5}
PEGEM^{1}
EADSIM^{1}
C4I^{1}
1-TSIU
2-Eagle/MCS
3-MAI
4-VHMS
5-CFOR
Division: MCS^{1}, ASAS-RWS^{1} FAAD^{1}, AFATDS^{1}
Brigade: MCS^{1,2}, ASAS-RWS^{1}, FAAD^{1}, AFATDS^{1}, MSAC
Battalion: Applique^{1,3}, MCS^{1,2,5}, ASAS-RWS^{1}, CSSCS^{4}, FAAD^{1}, AFATDS^{1}
Company Applique^{1,3,4}
Platoon Applique^{1,3}
Exp 3: DBBL - LW to Appliqué
M2
DI SAF
LW SVS
LW ODT
AI
Platoon: M2 Applique
Squad: Land Warrior System
Individual Soldier: Land Warrior System

Note: The coding scheme (i.e., 1, 2, 3, etc) is used to associate the simulation drivers, the interfaces, and the tactical systems (e.g., FIRESTORM feeds the ACE/ACT which feeds the Division and Brigade level ASAS-RWSs, etc).

Figure 2 -- AE4 architecture diagram for CBS, CLCGF, and DBBL experiments

Observations and Results

Run Time Manager (RTM)

The RTM addresses the challenge of providing a realistic linkage between CBS and Forward Area Air Defense (FAAD) systems as a potential alternative to the legacy Simulation Support Modules (SSMs). The RTM provides this linkage with less overhead (as compared to the SSMs), and provides a more realistic training environment due to the transmittal of perceived data versus ground truth data (through modeling of communications links via sensor models).

The RTM is a United States Army Space and Missile Defense Command (USASMDC) product developed by Booz-Allen Hamilton. The RTM provides a "two way" interface between aggregate level simulations (e.g., CBS for the AE4 CBS experiment) and entity level simulations (e.g., Extended Air Defense Simulation (EADSIM) for the AE4 CBS experiment), providing Aviation and Air Defense data to Forward Area Air Defense (FAAD) systems to include: Low Altitude Air

Picture (F-3 messages), Theater Ballistic Missile (TBM) Early Warning (F59-63 Messages) via FAAD Data Link (FDL), and High Altitude Air Picture via Tactical Digital Information Link-Broadcast (TADIL-B) (M0, M2, & M82).

Activity generated in EADSIM, such as aircraft detection by air defense radar, is translated into the appropriate tactical message formats and provided to FAAD systems at division and brigade levels. During the AE4 CBS experiment, the RTM provided a link from CBS to EADSIM, creating a high fidelity entity-based environment (in EADSIM) from aggregate air/ missile defense objects modeled and ghosted in CBS.

The RTM operates as follows: The aggregate level simulation acts as the controlling simulation over the entity level simulation. All unit attributes (speed, direction, formation, etc.) and engagement results are obtained from the aggregate level simulation and passed on to the entity level simulation. The RTM accomplishes the "cause-effect" relationship by utilizing Simulation Management (SIMAN) Protocol Data Units (PDUs) (e.g., "Stop", "Freeze", etc.) to control the entity level simulation while obtaining data from CBS.

In addition to CBS/EADSIM/FAAD, the RTM operates with the Air Warfare Simulation (AWSIM - Air Force aggregated simulation model), and the Joint Maritime Command Information System (JMCIS) which is one of the primary tactical C2 systems being used by the Navy.

Modular Reconfigurable C4I Interface (MRCI)

The MRCI adresses the challenge of providing a "proof of concept" linkage between CBS and Maneuver Control Systems (MCS) as a potential alternative to the legacy SSMs. However, a significant amount of development will be required before the MRCI can serve as a viable alternative to the SSM for linking CBS to MCSs.

The MRCI was part of a prototyping effort sponsored by the Defense Modeling and Simulation Office (DMSO), and was developed by Science Applications International Corporation (SAIC). The MRCI resided between the target live C4I systems, and the High Level Architecture (HLA) Run Time Infrastructure (RTI).

It is noted that due to the aggressive AE4 schedule and other issues, a "one way" implementation of the MRCI was performed only for position tracking of CBS units.

FIRESTORM/HRSS

The Federation of Intelligence, Reconnaissance, Surveillance and Targeting, Operations, and Research Models (FIRESTORM)/High Resolution Simulation Stimulator (HRSS) simulation addresses the challenge of providing an intelligence training simulation which can take feeds from CBS and provide perceived data to All Source Analysis Systems (ASASs). FIRESTORM provides a realistic training environment due to transmittal of perceived data (through modeling of communications links via sensor models).

FIRESTORM/HRSS is an interactive training system which utilizes an artificial intelligence implementation to challenge friendly force commanders. FIRESTORM continuously reacts to battlefield events, and attempts to take advantage of given tactical situations. FIRESTORM provides the simulation models required to provide data to the All Source Analysis Systems (ASAS) contained within the Analysis Control Element (ACE) to include: Image Intel (IMINT), Signal Intel (SIGINT) - consists of Electronics Intel (ELINT) and Communications Intel (COMINT), Human Intel (HUMINT) - provides Surveillance, Size, Activity, Location, Unit, Time, & Equipment (SALUTE) reports and Spot reports, and Rotary Wing Aircraft (RWA) models - provide reports similar to HUMINT.

The FIRESTORM simulation models currently include the Unmanned Aerial Vehicle (UAV), Joint Surveillance Target Attack Radar System (JSTARS) Simulator (JSS), Multiple UAVs in a Simulation Environment (MUSE), Advanced Synthetic Aperture Radar System (ASARS), Ground Based Common Sensors (GBCS), Common Ground Station Surrogates (CGS(S)), and Intelligent Minefields (IMF). The HRSS provides deaggregated representation of entities moving on the terrain so battlefield sensors can be used to detect them (resulting in perceived information).

FIRESTORM can be used with CBS, Janus, Distributed Interactive Simulation (DIS), and other Joint Training Confederation (JTC) simulations to provide a realistic ISR environment.

It was noted during the AE4 CBS experiment that FIRESTORM overwhelmingly defeated friendly forces contained within CBS. This was attributed to the use of a scripted scenario for the AE4 CBS experiment. FIRESTORM was designed as a training simulation to "punish" friendly force commanders who do not effectively take advantage of tactical situations as they arise during scenario execution. Thus, FIRESTORM requires dynamic interaction by friendly force commanders. The use FIRESTORM/HRSS with a scripted scenario (such as the one used to support the AE4 experiments) requires constant monitoring to ensure that appropriate adjustments are made.

Simulation Support Modules (SSMs)

The SSMs are legacy interfaces that were developed by TRADOC Test and Experimentation Command (TEXCOM) and the Electronic Proving Ground (EPG). They are the primary interfaces currently used for providing the linkages between CBS and the Army Tactical Command and Control System (ATCCS). Initially, the SSMs were designed to support technical and operational testing, but have recently been used to support training exercises as well (e.g. Division XXI). It is noted that the SSMs (in conjunction with CBS) provide "ground truth" data from CBS to the target C4I systems, and not "perceived' data as was the case with some of the other simulations/interfaces discussed in this document.

Tactical Simulation Interface Unit (TSIU)

The TSIU addresses the challenge of providing realistic linkages between Distributed Interactive Simulations (DIS) and the ABCSs from Division level all the way down to the platform level. The newer features added to the TSIU via the AE4 experiments were the "two way" command and control linkages to Force Battle Command Brigade and Below (FBCB2) systems (formerly referred to as Applique). The TSIU provides a realistic training environment due to the transmittal of perceived data versus ground truth data (through modeling of communications links via sensor models).

The TSIU is a USASMDC product being developed by Coleman Research Corporation. The TSIU is an interface which translates data between aggregate/entity level Distributed Interactive Simulations (DIS) (such as Eagle and Modular Semi-Automated Forces (ModSAF)), and ABCSs utilizing a wide variety of message protocols. The TSIU is used in conjunction with simulations such as EADSIM and the C4I surrogate simulation to realistically model C2 messaging based on perceived information rather than ground truth information.

The AE4 CLCGF Experiment provided a means to evaluate the performance of the TSIU in utilizing aggregated (Eagle) and deaggregated (entity level) simulation data. This interface provided simultaneous multi-level (i.e., division, brigade, battalion, company, and platoon) feeds to tactical C2 systems, as well as a "two way" VMF interface between FBCB2 systems and simulations.

One of the primary reasons the TSIU was selected for participation in the AE4 was its robust capability for stimulating multi-echelon ATCCS systems. The TSIU was enhanced for the AE4 to provide an interface between simulations and FBCB2 systems. This was envisioned to be a worthwhile investment because the entire functionality of the TSIU would be resident within a single Sun SPARC 20/Ultra SPARC workstation, resulting in minimal hardware requirements and contractor support costs for supporting exercises.

Currently most of the data flow though the TSIU is "one way" from simulations to the ABCSs. As more simulations are developed such that they can accept "input" data from C4I systems and react appropriately, the TSIU can be enhanced to provide the required "two-way" information flow. A proof of concept for "two-way" information flow was demonstrated during the AE4 CLCGF experiment, in that enhancements were made to the TSIU for translation of simulation data to Variable Message Format (VMF) data (for communication with the FBCB2 systems) and vice versa. VMF data was provided to a developmental version of ModSAF containing the CCSIL libraries (referred to here as the Command Forces (CFOR) ModSAF, but also commonly referred to as the STOW '97 version of ModSAF) for execution of an automated "call for fire" function. In addition, friendly and perceived enemy icons appeared on the appropriate FBCB2 screens as generated by the driving simulations. The TSIU provides the capability to pass information from the simulation environment to the target C4I systems to include: blue force tracking (United States Message Text Format (USMTF) S507, VMF K5.01, CCSIL 201), high and low altitude air picture (TADIL-B, FDL), early warning Nuclear, Biological, Chemical (NBC) (NBC-1,4, USMTF C488, VMF K5.61), early warning Theater Ballistic Missile (TBM) (predicted impact point – FDL F-59, FDL F-60, FDL F-61, FDL F-62, FDL F-63, USMTF C121), intelligence data (Spot reports via VMF K4.52), Radar Exploitation Reports (REXREPs), SALUTE reports, red order of battle, JSTARS Moving Target Indicator (MTI), telemetry (UAV/JSTARS position), TBM (estimated launch point, pairing with impact point), Tactical Electronics Intelligence (TACELINT), and ACE (USMTF S309)), and C2 messaging (USMTF, VMF) to include: call for fire (VMF K2.04, CCSIL 401) and status reporting.

The primary TSIU enhancements performed in support of the AE4 were the following:

1. Stimulation of target FBCB2 systems with simulation data -- The TSIU was enhanced to provide a translation of CCSIL to Variable Message Format (VMF) messages so that "perceived" simulation data could be transmitted to the Company level and platoon level FBCB2 systems, allowing Blue Force Tracking and monitoring of Spot Reports by operators using the target FBCB2 systems. This resulted in friendly and perceived enemy forces to be displayed upon the FBCB2 screens.

2. "Call For Fire" Function -- The TSIU was enhanced to provide a translation of VMF to CCSIL messages to facilitate an automated "Call For Fire" function from given FBCB2 systems to CFOR ModSAF. CFOR ModSAF, however, did not react to the message. This was viewed to be a limitation within CFOR ModSAF (a developmental version).

The logged simulation data streams from the AE4 CLCGF experiment were used to fully stimulate (through the TSIU) the ABCSs at each level of command for display at the Association of the United States Army (AUSA) convention 13-15 Oct 97 in D.C. This required compressing the three and a half hour logged experiment scenario into a twenty minute presentation. This demonstrated that the TSIU could be used with a data logger system to support robust After Action Review capabilities.

The TSIU has been successfully utilized in experiments of the past to include: AE2, AE3, Digital Training Exercise (DTX) - Fort Rucker, and Roving Sands Experiment - White Sands Missile Range, Battlespace Integration Concept Emulation Program (BICEP), and is currently a Warfighting Rapid Acquisition Program (WRAP) candidate.

MITRE Appliqué Interface (MAI)

The MAI addresses the challenge of providing realistic "two way" linkages between DIS simulations, and FBCB2, using the HLA. The MAI provides a realistic training environment due to the transmittal of perceived data versus ground truth data (through the modeling of communications effects).

The MAI was the result of an internal research and development effort performed by MITRE Corporation. The MAI provided a "two way" interface between FBCB2 systems and simulations. The MAI development effort examined the potential of the HLA to provide a framework for integrating C4I systems with simulations. The baseline capability was extended by introducing realistic communications effects to C2 message traffic between /among simulation and C4I systems using the HLA/RTI.

The MAI provided friendly force tracking, enemy force tracking (via spot reports), and battle damage assessments data to FBCB2 systems. The following automated feature was also accommodated via this interface: the translation of VMF free text Movement Orders into CCSIL using the MAI, so that movement could be automatically executed by CFOR ModSAF.

The MAI experiment set-up included a company and platoon level FBCB2 system, and was conducted in two phases. During the first phase, the MAI functionality was demonstrated using the Fort Hood database which contained an artificial obstruction to block Line Of Sight (LOS) communications between C2 vehicles. During the second phase, the MAI functionality was demonstrated using the natural terrain obstructions contained within the Grafenfels database. The following was observed during these phases of the AE4 CLCGF experiment:

1. Entity state Protocol Data Units (PDUs) provided own vehicle locations (as well as all other vehicle locations), thus simulating the effect of the Global Positioning System (GPS).

2. VMF free text messages issued by the FBCB2 systems were translated into CCSIL execute directives, and used to modify CFOR ModSAF entity tasking. Specific identifiers contained within the free text allowed the performance of this function. This functionality was demonstrated for obtaining status reports from given CFOR ModSAF entities (e.g., mission capability, available supplies, etc.), and task these entities to change their paths.

3. All messages passed between CFOR ModSAF and the FBCB2 systems were transported through the HLA/RTI. The HLA implementation utilized the RTI version 1.0.2, and was comprised of the following three federates:

   a. Communications Effects Federate – Consisted of a propagation model (based on the Terrain Integrated Rough Earth Model (TIREM) model which calculated the propagation/degradation of message traffic over terrain, taking into consideration Line Of Sight (LOS), and external effects such as jamming, and interference), a VMF to CCSIL translator, a CCSIL to VMF translator, and a position Situational Awareness (SA) server (used to provide data updates to target FBCB2 systems on a periodic basis when specified minimum threshold values were exceeded for time between updates and distance traveled).

b. C4I Federate: Designed to perform the general integration of C4I command and control systems. For this particular experiment, two FBCB2 systems represented this federate.

c. Simulation Federate: Designed to perform the general integration of simulation systems. For this particular experiment, CFOR ModSAF represented this federate.

Land Warrior (LW) Interface

The LW Interface addresses the challenge of providing a live C4I linkage between the LW Command and Control Systems and the FBCB2 C2 system. The LW system is a prototype developed for use by the Dismounted Infantry community. The LW C2 and FBCB2 linkage provides a proof of concept for allowing dismounted soldiers to communicate with higher headquarters via FBCB2.

The Land Warrior Interface was developed by Hughes Aircraft Corporation. The LW interface system provided the VMF conversions required for supporting the "two way" LW C2 system linkage to FBCB2. Spot reports were sent from the LW system to the FBCB2 system resulting in enemy icons being displayed on the target FBCB2 system. As entities populated the FBCB2 system, these were displayed within the LW Integrated Helmet Assembly Subsystem (IHAS) as overlays.

Medical Situational Awareness and Control (MSAC) System

The MSAC system addressed the challenge of providing a realistic C2 system for tactical medical operations, and represents the first time the medical community has been able to actively participate in a simulation exercise in a distributed fashion (up to this point medical simulation had only been performed in a "stand-alone" mode).

The MSAC system was developed for the United States Army Medical Research Materiel Command (USAMRMC) and the Army Medical Department (AMEDD) Center & School by Mystech. The MSAC serves as a C2 system for tactical medical operations, providing total situational awareness for friendly and enemy forces/objects such that Line and Medical Commanders are challenged to make "real time" battlefield decisions with respect to route planning considerations, selection of available evacuation platforms and treatment facilities, and in providing medical logistics. A casualty generator is used to provide notification regarding casualties. As casualties are incurred, the medical support team executes their mission taking into account the data received by the MSAC system. MSAC provides a map-based display with overlays, symbology, and other graphics capabilities to present red and blue force orders of battle, terrain details, routes, landing zones, and other data critical to line and medical commanders. Using a Netscape implementation, MSAC allows users to click on various map icons to display database records and reports for selected functional areas.

Currently the MSAC has been fully integrated with the MCS/Phoenix (P) beta version which provides the required situation awareness data. For the AE4, portions of the MSAC system were integrated with the MCS baseline version as a proof of concept. In the future it is anticipated MSAC will be integrated into the Combat Service Support Control System (CSSCS).

Vehicle Health Monitoring System (VHMS)

The VHMS system addresses the challenge of enhancing simulation with respect to the modeling of logistics functions, and populating FBCB2/CSSCS systems with this data.

The VHMS was developed by the Aviation and Missile Command (AMCOM) Logistics Laboratory. The VHMS consists of the following components:

1. VHMS module -- This component is mounted on a live military vehicle, and is used to collect and transmit data on system performance to include: fuel & ammunition levels, and critical engine and suspension components. This data is transmitted to a remote VHMS workstation.

2. VHMS workstation -- This component is used to receive live data from the VHMS module, model the functionality of the VHMS for simulated entities, and transmit digital logistics data received through a target FBCB2 system to CSSCS.

During the AE4 CLCGF experiment, the VHMS workstation combined the data received from the VHMS simulation and live VHMS module, and successfully populated the target FBCB2 system with the required information. The FBCB2 system performed a "roll-up" of the data received so that this could be passed on to the CSSCS. It is noted however, that the linkage between FBCB2 and the CSSCS was not established due to database correlation problems. This was not a problem within the VHMS, but the target C4I systems, and is one example of the interoperability problems that can exist among ABCS systems.

Additional information concerning the VHMS can be found at the following web site address: http://loglab.redstone.army.mil

Other Systems

Information regarding other systems that participated in the AE4 experiments are provided in the references listed at the end of this paper[1,2,3,4].

Conclusions and Recommendations

Several conclusions and recommendations have resulted from the execution of the AE4 experiments. Some of these are discussed below:

1. Perceived versus Ground Truth data. It is important to define the distinction between perceived and ground truth data. Perceived data creates the "fog of war" effect which results when sensors provide information which may not be entirely accurate (as would be the case with ground truth data), or limited to no information when sensor systems become damaged. Perceived data better accommodates training applications (as compared to ground truth data) for:

   a. trainees will be challenged to take measures for ensuring battlefield sensor systems are adequately protected.

   b. trainees will be challenged to weigh tradeoffs (potential threat versus benefit gained) prior to committing battlefield sensors for specific missions.

   c. given that specific battlefield sensor information is not received, trainees will be challenged to troubleshoot the networks to determine the cause (e.g., damage to sensor system, network problem, etc.) and take corrective action as required.

The use of ground truth data provides strong potential for a negative training effect because:

   a. trainees would be receiving more accurate information than would be the case in an actual combat scenario.

   b. there would not be a need to protect sensor assets (ground truth data would be received whether sensors were damaged or not).

   c. trainees would be spending more time focusing on the mission, and less time on trouble shooting and verification/validation as would be the case in an actual battlefield scenario.

3. Perceived data can be accommodated the following ways:

   a. providing adequate models of battlefield sensors and communications links to provide perceived information regarding the entities detected. During the AE4 experiments the following simulation/interface combinations provided perceived data in this manner: TSIU/EADSIM/ C4I Surrogate/CFOR ModSAF, FIRESTORM, and RTM/EADSIM.

   b. providing adequate modeling of communications effects. During the AE4 experiments the following simulation/interface combination provided perceived data in this manner: CFOR ModSAF/MAI/CES.

4. C4I/Simulation-based laboratory benefits. There exists significant benefit for utilizing a simulation based laboratory to provide a controlled environment for testing ABCS interoperability, performing human factors analysis studies, experimentation with visual/audio cues and information filtering, prototyping new systems, optimizing system designs, and for general debugging purposes. Currently due to the level of sophistication of the ABCSs, a significant amount of initial and reinforcement training is required to maintain an adequate level of operator expertise, thus enhancements to the ABCSs (and other C4I systems) in some of the areas mentioned above could offer a significant benefit.

5. Designing C4I systems and associated training as one system. Because the use of simulation is the most effective means of providing battlefield conditions/data for stimulation of C4I systems in a training environment, it is recommended that C4I systems and other fielded systems be designed with this in mind (i.e., the fielded system and training be considered as one complete system). The merging

of these communities would better allow appropriate tradeoffs to be made in the design of fielded systems with respect to training considerations, and better accommodate the objectives for embedded training. By optimizing fielded C4I systems and associated training systems as a whole, overall design constraints would become more apparent. These design constraints could then be communicated to the development community to ensure effort is not expended on the development of training solutions which are not considered feasible. Note: It was speculated during the AE4 experiments that the loading of software onto target C4I systems for training purposes was not acceptable, however, three of the approaches evaluated relied on this approach (i.e., Eagle-MCS, MRCI, and MSAC). It is unknown how the objectives for embedded training can be accommodated without allowing the loading of software onto target C4I systems.

Current Challenges and Potential Areas of Follow–on Work

Based on the observations made during the AE4 experiments, the following are current challenges and some areas of "follow-on" work which offers potential benefit to the simulation/C4I community:

1. Synthetic Environment Representation for C4I. A significant amount of enhancements to the synthetic environment are required to facilitate ABCS training. Constructive aggregate level simulations (such as CBS) would better accommodate training requirements for modern C4I systems if they could track units all the way down to the entity level. This would prevent some of the data overload problems experienced during the AE4 and other similar experiments, and offer additional flexibility for C4I system Commanders.

It is noted that the STRICOM ADST II team will be utilizing their simulation/C4I/HLA testbed to perform additional experiments to investigate simulation/C4I related issues, assist in determining/coordinating C4I related requirements for synthetic environments, and provide a means to evaluate/test ongoing synthetic environment enhancements.

2. Alternative interfaces to the SSMs. One of the experimental objectives for the AE4 CBS experiment was to evaluate alternatives to the SSMs for linking CBS to the ATCCS. The primary concern with the SSMs is that development and maintenance can be a tedious and expensive procedure, and a large amount of exercise control staffs and support personnel are required to support large scale exercises. The MRCI provided a proof of concept for linking CBS to MCSs via the HLA, but only provided position tracking of the units contained in CBS. The RTM provided a robust interface between CBS/EADSIM and FAAD systems, but will require additional development for providing linkages to the other ATCCS. The National Simulation Center (NSC) is currently performing additional development/experiments with the RTM to determine it's feasibility for providing a linkage between CBS and other ABCSs (in addition to FAAD). It is noted that use of a similar approach to that provided during the AE4 CBS experiment (CBS combined with entity level simulations) for linking CBS to the remaining ATCCS components may not prove to be feasible (in particular when providing stimulation of the MCSs). This is primarily because, the amount of hardware and support personnel required to provide entity level representation of the aggregate forces contained within CBS (especially for the ground forces) could be substantial.

It is noted that the SSMs will continue to be required for supporting CBS exercises until robust alternatives to these interfaces become available.

3. Impact of HLA. Additional experiments are required to examine the impact of HLA with respect to legacy simulations, and interfaces between these simulations and C4I systems. Currently there are a number of alternatives available to move legacy ALSP-based and DIS-based simulations towards HLA. However, the interfaces that facilitate intercommunication with the C4I systems will also require examination in this regard. The STRICOM/LMC ADST II team is currently planning to team with the SMDC team to investigate these issues using the ADST II simulation/C4I/HLA Test Bed.

4. Enhancing the TSIU with communications effects. As a result of the AE4 experiments, it was determined that the TSIU was one of the more advanced interfaces between simulations and C4I systems, and that it offered a very robust capability for stimulating ABCS systems from Division all the way down to the platform level. The TSIU (in conjunction with EADSIM and the C4I Surrogate simulations) provided perceived data by modeling the sensor models and the associated communications links. To gain a better representation of perceived data, it would be

beneficial to integrate a model which simulates network and communications propagation effects. The STRICOM/LMC ADST II team is currently proposing to team with SMDC to integrate the Situational Awareness Tactical Internet Data Server (SATIDS) functionality as part of the TSIU thus, providing a more robust capability that can be used in the field for training purposes. This would provide tactical internet modeling and communications effects for the TSIU.

5. Training using ABCS systems. The theme approved by the Chief of Staff of the Army for AE5 is "Training Army XXI Leaders to Exploit Situational Awareness". This is a paradigm shift from previous AEs which centered on technology. The objectives of AE5 are: To present existing and emerging training capabilities; Support the development of a Digitized Leader Reaction Course (DLRC) and execution drills; Identify methods in optimizing the management of battle command information and maximizing the use of situational awareness; Support the implementation of emerging training strategies for Army XXI; Enhance training and training support systems for leaders of digitized units; and Leverage "Early User Capabilities" of emerging training systems. To meet these objectives, AE5 plans a four phase approach. The first phase will involve conducting a Digitized Training Experiment (DTE) at the Warlab, in Fort Leavenworth, KS. This experiment will support the DLRC and leader execution drills development, evaluate battle command information management tools and use of situational awareness, assess the three step process as a way to train Army XXI leaders, and assess the effectiveness of After Action Review (AAR) systems as training support tools for digitized leaders. The second phase will involve conducting a Digitized Training Exercise (DTX) at Fort Hood, TX. This exercise will support the train-up of the 1st Brigade of the 4th Infantry Division (ID) for the FBCB2 Limited User Test (LUT) by transporting capabilities to the user, test the AE5 hypotheses, support the implementation of emerging training strategies for Army XXI, enhance training and training support systems for leaders of digitized units, and leverage early user capabilities of emerging training systems. The third phase will involve a collateral data collection effort. The objective of this effort is to determine the effectiveness of the Force XXI Test Procedures, DLRC, and the three-step training process in preparing leaders to exploit situational awareness. The fourth phase will involve the actual presentation at the AUSA conference in October 1998. The presentation will capture the essence of the other three phases and showcase them.

Residuals and Recipients

The residual components discussed in this document are identified within two of the references listed at the end of this paper[3,4].

Participating Organizations

The success experienced on the AE4 program was largely due to the strong teamwork among the organizations involved. The organizations which supported the AE4 team were comprised of several Government and Contractor organizations. These organizaitons are identified in the references listed below[1,2,3,4].

References

1. Joseph Brennan, Kent Bimson, and Mike Kalaf: "The Army Experiment 4 (AE4) Simulation to Army Battle Command System (ABCS) Experiments" (Fall '97 SIW paper), June 1997.

2. STRICOM and the ADST II AE4 contractor support team: Technical Concept For Army Experiment IV (AE4) DO 0034 - CDRL AB02", 29 August 1997.

3. Joseph Brennan: "The Army Experiment 4 (AE4) Simulation to Army Battle Command System Experiment Results (Spring '98 SIW paper), Jan 1998.

4. STRICOM and the ADST II contractor support team: "Army Experiment 4 After Action Report" CDRL, May 98.

Author Biographies

Mr. Joseph M. Brennan is a Systems Engineer at United States Army Simulation, Training, and Instrumentation Command (STRICOM) in Orlando, Florida who supports the Advanced Distributed Simulation Technology (ADST) II contract. Mr. Brennan earned a Master's of Science degree in Industrial Engineering from Texas A&M University, and a Bachelor's of Science in Mechanical Engineering from University of Central Florida. Mr. Brennan's research interests currently include: the study of linkages between simulations and command and control systems, High Level Architecture (HLA), and Synthetic

Environment Data Representation and Interchange Specification (SEDRIS).

Ms. Rita Simons is a Systems Engineer at United States Army Simulation, Training, and Instrumentation Command (STRICOM) in Orlando, Florida who supports the Advanced Distributed Simulation Technology (ADST) II contract. Ms. Simons earned a Bachelor's of Science degree in Electrical Engineering from the University of Central Florida and is currently pursuing her Master's of Science degree in Simulation at the University of Central Florida. Her research interests currently include C4I interfaces, HLA, and the use of unmanned vehicles to augment the mounted force.

AIAA-98-4171

TACTILE CUEING MODEL FOR G-SEAT USE IN FLIGHT SIMULATION

Patricia B. Schmidt and Laurence R. Young, Sc.D.

Massachusetts Institute of Technology, Cambridge, MA

Abstract

Tactile cueing using G-seats has been used to provide normal acceleration and roll and pitch tilt cues in fixed and moving base flight simulators. Although the addition of tactile cueing has been shown to improve pilot performance in fixed base simulation, its transfer value and training benefit remain unclear. A G-seat drive algorithm based on a validated model of motion perception from tactile cues is likely to be more useful in training than a drive algorithm that is based solely on simulator pilot performance. The contribution of tactile cueing to motion perception is less well understood than the contribution of visual and vestibular cues to motion perception. An experiment was conducted to quantify the frequency dependent contribution of tactile cueing to linear motion perception using a modified NASA Langley Research Center pneumatic G-seat. Eight blindfolded subjects participated in a horizontal linear motion nulling experiment, lying on their backs during z-axis motion on the MIT sled. Sum of sines disturbances (.06 Hz-.5 Hz) were provided in both sled velocity and G-seat pressure in one half of the experimental trials and only sled velocity in the other half of the trials. Subjects attempted to null the sled velocity using a hand controller. Transfer functions from sled velocity to subject command response (vestibular transfer function) and from G-seat pressure to subject command response (tactile transfer function) were computed using cross-correlation methods for each of the two conditions (G-seat on and G-seat off). A significant response to G-seat pressure, fit with a differentiator transfer function, was observed. Differences between the G-seat on and G-seat off vestibular transfer function were significant but smaller than expected. Differentiation in the tactile transfer function agrees with previous research on tactile receptors in the skin and supports the use of G-seats to cue acceleration onset rather than acceleration magnitude. A normal acceleration based drive algorithm for a pneumatic G-seat and a simulation concept using a pneumatic G-seat and a helmet-mounted display are proposed.

Introduction

Motivation

Research on the use of tactile cueing in flight simulation has shown that tactile cues play a role in motion perception.[2, 6] However, tactile cueing has not been incorporated into an overall model of spatial orientation. Consequently, tactile cueing drive algorithms, with some exceptions, have relied on pilot performance in the simulator as a criterion for evaluating the realism of the tactile cueing scheme.[4, 6]

Experimentation outside of the flight simulation environment contributes to the development of tactile cueing devices by isolating the contribution of tactile cueing to motion perception and allowing the perceptual response to tactile cueing to be measured. Incorporation of tactile cueing into a model of human spatial orientation would allow more realistic G-seat drive algorithms to be designed, increasing the realism and training value of G-seats.

Background

A G-seat is a flight simulator pilot's seat whose shape and firmness characteristics can be varied, in order to mimic the tactile sensations produced by contact between the pilot's body and seat in an accelerating aircraft.

Two types of G-seats have been used for the majority of tactile cueing research in flight simulators. The pneumatic G-seat, developed at NASA Langley Research Center, has eight inflatable bladders on the seat pan and back. The pressure in each of the inflatable bladders can be varied independently, in order to alter the firmness of the seat. The Langley G-seat was originally intended to add long-duration z-axis acceleration cues to full-motion simulation, and has been used in motion-base simulation.[1, 2]

The Advanced Low Cost G-Cueing System (ALCOGS) was developed for the U.S. Air Force. The ALCOGS has an array of rigid panels in the seat pan and back that are hydraulically actuated to control the shape of the seat. It was investigated as a low-cost substitute for simulator motion in cueing roll and pitch tilt.[5]

Previous Research

Research on tactile cueing in flight simulation has focused on pilot performance in tracking tasks using G-seats and has evaluated G-seat effectiveness and the realism of various drive algorithms.

Ashworth et al. demonstrated that a pneumatic G-seat used to cue z-axis acceleration improved pilot performance in full-motion simulation.[2] Martin et al. likewise showed a performance improvement in fixed base simulation using the ALCOGS, but did not demonstrate a training benefit of the ALCOGS when pilots were later tested in full-motion simulation.[6]

Research on tactile receptors in the skin has focused on the responses of individual receptor units, rather than responses to the large-area stimuli that result from contact between a pilot's body and the seat. Recordings from individual receptors have shown that the tactile receptors transduce a weighted sum of velocity and position of skin indentation, rather than position of skin indentation alone.[3]

Objective

The objective of this study was to quantify, in the frequency domain, the contribution of G-seat tactile cueing to linear z-axis motion perception. Two questions were posed.

- Is there a measurable contribution of G-seat tactile cueing to motion perception?
- What is the dynamic relationship between G-seat pressure and linear motion perception?

Methods

Experiment

Eight blindfolded subjects participated in a z-axis linear motion nulling experiment. They were presented with tactile cues from the seat pan portion of a Langley pneumatic G-seat and linear horizontal motion from the MIT sled. Subjects were seated on their backs, with legs raised, on the sled, as shown in Figure 1. Sled motion was horizontal and along the subject's z-axis.

Subjects attempted to null sled velocity using a hand controller that commanded sled velocity, while pseudorandom disturbances were given in sled velocity and G-seat pressure in half of the experimental trials (G-seat-on trials) and only sled velocity in the remaining trials (G-seat-off trials). Disturbances were uncorrelated sum-of-sines containing 12 frequencies between 0.06 Hz and 0.5 Hz. The subjects were blindfolded and sounds produced by the sled were masked by broadband noise from an AM radio tuned off of a station.

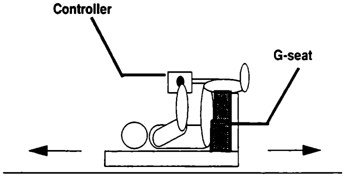


Figure 1. Experiment configuration.

Analysis

The subject's command signal, sled velocity, and G-seat pressure were recorded. Two transfer functions were calculated for each trial, using cross-correlation methods.[9] For the G-seat-on trials, the tactile transfer function, from G-seat pressure to velocity commanded by the subject, and the vestibular transfer function, from sled velocity to velocity commanded by the subject, were calculated. For G-seat-off trials, the vestibular transfer function and the remnant magnitude were calculated. The remnant is defined as the subject's response that is not correlated with the inputs.[7]

Remnant magnitude was compared to the tactile transfer function, in order to determine whether the computed tactile transfer function magnitude was significantly greater than the remnant magnitude. Satterwaite's method for assessing significance of differences between data of different variances was used to evaluate the significance of differences between the tactile transfer function magnitude and the remnant magnitude.[8]

A transfer function was fit to the calculated tactile transfer function data. Since only magnitude data was used in the transfer function fit, an iterative search method was used to find the pole and zero locations which fit the data best in a least-squares sense. Goodness of fit was assessed with an F-test method for multiple linear regressions.[8]

Results

Response to G-seat

The mean tactile transfer function magnitude was significantly greater than the mean remnant magnitude at all of the frequencies tested. Figure 2 shows the comparison between the mean tactile transfer function and the mean remnant magnitude. Individual subjects' tactile transfer function magnitudes were significantly greater than remnant magnitudes at 8 of the 12 frequencies tested for 7 of the 8 subjects. The lack of significant differences for the remaining subject can be explained by the loss of a portion of that subject's data due to a computer error.

Tactile transfer function

The transfer function that best fit the tactile transfer function data in a least-squares sense had a single zero at a frequency near the lowest frequency used in the disturbance. The tactile transfer function data and the fitted transfer function are shown in Figure 3. Although the frequency of the zero can not be determined from this data, the tactile transfer function data is consistent with a differentiator transfer function over the range of frequencies tested (0.06 Hz-0.5 Hz). The differentiator tactile transfer function is given below, where V is the velocity commanded by the subject, P is the G-seat pressure and s is the Laplace variable. The gain is in arbitrary units ((m/s)/kPa).

$$\frac{V(s)}{P(s)} = 0.690\,s$$

Discussion

Response to G-seat

The finding that tactile transfer function magnitudes were significantly higher than remnant magnitude indicates that the G-seat tactile cueing did have an effect on linear motion perception.

The computed tactile transfer function was best fit by a differentiator. This indicates that, over the fre-

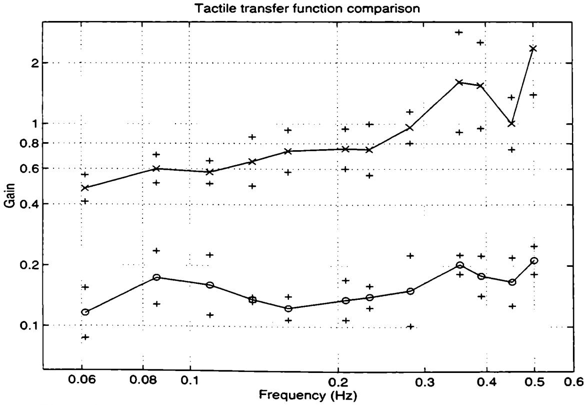


Figure 2. Tactile Transfer Function Comparison. x indicates tactile transfer function, o indicates remnant magnitude, + indicates mean +/- 1 standard deviation.

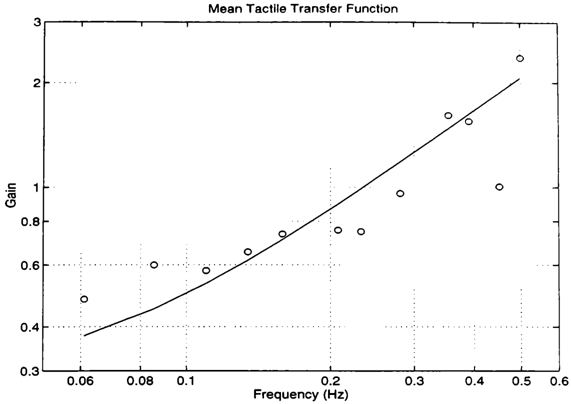


Figure 3. Fitted Tactile transfer function. o indicates data, line indicates fitted transfer function.

quency range tested, subjects were responding to the time rate of change of G-seat pressure, rather than the magnitude of the pressure. The differentiator tactile transfer function agrees with research on individual tactile receptors, which indicates that the receptors' responses depend on the velocity of skin indentation.

The differentiator transfer function lends support to the concept of using G-seats to cue onset and changes in acceleration, rather than acceleration magnitude. Pneumatic G-seats used to cue z-axis acceleration are driven so that G-seat pressure is proportional to the simulated aircraft's z-axis acceleration. The finding that subjects responded to the time rate of change of G-seat pressure, rather than its static value, suggests that the information pilots receive from tactile cueing is the rate of change of acceleration, rather than acceleration magnitude.

Seat firmness and seat position cueing

In a real aircraft, $+G_z$ (downward) acceleration results in an increase in apparent seat firmness, accompanied by a downward shift in eye position, as the pilot sinks into the seat. Neither the pneumatic G-seat nor the ALCOGS reproduce both of these cues in synchrony.

The pneumatic G-seat has a non-monotonic relationship between seat firmness and seat height. Over a limited range of low pressures, the height-firmness relationship corresponds with that of a real aircraft seat. An increase in G-seat pressure raises the eyepoint slightly and causes the pilot's weight to be supported by the inflatable cushions, decreasing seat firmness, while a decrease in pressure lowers the eyepoint and places the pilot's weight on the hard seat base.[2] At higher pressures, the height-firmness relationship reverses. Increasing the G-seat pressure raises the eyepoint, while making the seat firmer. Although operating the pneumatic G-seat in the low pressure range allows for the correct polarity in the relationship between seat height and seat firmness, the seat height changes are small, and when the G-seat pressure exceeds the limits of the low pressure operating range, confusing false tactile cues result.

The ALCOGS was designed to have inflatable firmness bladders overlying the rigid seat panels, in an effort to independently control seat height and firmness.[5] However, inertia of the pilot's body causes changes in apparent seat firmness which are counter to those experienced in real aircraft. A rapid downward motion of the seat pan, used to cue downward acceleration, results in both a downward shift in eye position and a transient decrease in the apparent firmness of the seat, as the pilot's body lags the seat's vertical motion.

Flight simulation using a virtual reality helmet mounted display could combine the seat firmness cueing of a pneumatic G-seat operating in its high pressure range with a realistic shift in eyepoint position. If the cockpit interior is displayed in the helmet mounted display, the eyepoint position may be shifted independently of other simulator devices. For $+G_z$ (downward) accelerations, the G-seat pressure is increased, making the seat more firm, while the eyepoint position in the display is lowered, resulting in a more realistic representation of the changes in seat firmness and eyepoint position caused by z-axis accelerations.

Conclusions

Although simulation studies have indicated that tactile cuing contributes to motion perception, the contribution of G-seat tactile cueing to linear motion perception has not previously been modeled. In this study, G-seat tactile cueing was demonstrated to have a measurable effect on linear motion perception. The dynamic relationship between G-seat pressure and perceived velocity was consistent with a differentiator, indicating that subjects responded to the time rate of change of G-seat pressure, rather than the magnitude of the G-seat pressure. This finding suggests that tactile cueing of z-axis acceleration provides pilots with information about the time rate of change of acceleration, supporting the use of G-seats to cue onset and changes in acceleration.

References

1. Ashworth, B. "A Seat Cushion to Provide Realistic Acceleration Cues for Aircraft Simulators," NASA TM X-73954, 1976
2. Ashworth, B., McKissick, B., Parrish, R., "Effects of Motion Base and g-Seat Cueing on Simulator Pilot Performance," NASA TP2247. March 1984.
3. Burgess, P. R. and Perl, E. R. "Cutaneous Mechanoreceptors and Nociceptors" in *Handbook of Sensory Physiology Vol. II: Somatosensory System.* edited by A. Iggo, Springer-Verlag, Berlin, 1973.
4. Flach, J., Riccio, G., McMillan, G., Warren, R. "Psychophysical Methods for Equating Performance Between Alternative Motion Simulators," *Ergonomics* Vol. 29, no 11, pp. 1423-1438, 1986.
5. Kron, G.J. and Kleinwaks, J. M. "Development of the Advanced G Cuing System." AIAA 78-1572., 1978.
6. Martin, E., Osgood, R., McMillan, G. "The Dynamic Seat as an Onset Cuing Device," AIAA 87-2438-CP. *AIAA Flight Simulation Technologies Conference*, Monterrey, CA, August, 1987.
7. McRuer, D., Clement, W., Thompson, P., Magdaleno, R. *Minimum Flying Qualities Volume II: Pilot Modeling for Flying Qualities Applications.* Systems Technology, Inc. 1989 Report No. 1235-1.
8. Rosner, B. *Fundamentals of Biostatistics,* 2nd Edition. PWS Publications, Boston, 1986.
9. Schmidt, P. *The Effect of G-Seat Tactile Cueing on Linear Motion Perception*, S.M. Thesis, Massachusetts Institute of Technology, 1998.

AIAA-98-4172

INVESTIGATION OF ROLL-LATERAL COORDINATED MOTION REQUIREMENTS WITH A CONVENTIONAL HEXAPOD MOTION PLATFORM

William W. Chung*
Logicon Syscon Inc., Syre
Moffett Field, California

Doug J. Robinson*, Jason Wong
Lockheed Martin Skunk Works
Palmdale, California

Duc Tran†
NASA Ames Research Center
Moffett Field, California

Abstract

This study is a follow-up to a series of investigations on the roll-lateral motion fidelity requirements based on a roll and lateral degree-of-freedom model with a desired roll rate command system in a sidestep task. In one of the previous studies, it was found that coordinated lateral motion was important for the roll motion. Otherwise, the uncoordinated specific force due to roll motion has an effect on the motion fidelity. In typical motion-based flight simulations, the coordinated lateral motion commands are being generated from a separate washout filter other than the roll axis to mitigate the displacement limitation. Therefore, pilot perceived specific force is determined by both the magnitude and phase characteristics of this coordinated lateral washout filter with respect to the roll motion. The magnitude dependent effects of the coordinated lateral motion have been investigated and criteria have been developed. This study focused on the phase dependent effect due to such applications. Results show that the overall motion fidelity is dependent on both the roll motion fidelity and the coordinated lateral motion fidelity. A general motion fidelity criterion is developed for a sidestep task.

Nomenclature

a_y	measured simulator lateral acceleration at the pilot station, y-body, ft/sec^2
g	gravitational acceleration, ft/sec^2
$L_{\delta_{lat}}$	helicopter roll control power, rad/sec^2/in
L_p	helicopter roll acceleration due to roll rate, 1/sec
p	helicopter roll angular rate, rad/sec
p_{cmd}	simulator roll rate motion command, deg/sec
$\dot{p}_{cmd}$	simulator roll angular acceleration command, deg/sec^2
r_z	vertical displacement between pilot abdomen and the simulator rotational center, positive downward, ft
v	helicopter translational velocity, y-body axis, ft/sec
$\dot{v}$	helicopter translational acceleration, y-body axis, ft/sec^2
$y_{a/c}$	helicopter lateral c.g. position, ft
y_{cmd}	simulator lateral travel command, ft
δ_{lat}	pilot lateral stick input, in.
ϕ	helicopter roll attitude, rad
ϕ_{cmd}	simulator roll attitude command, deg

Introduction

Motion systems have been commonly used to provide pilots an additional degree of realism in conducting ground-based flight activities to meet the mission requirements. However, since the travel of the motion-based flight simulators are significantly limited by physical space, determining how much motion cueing fidelity is required is difficult to characterize. The fact that motion has to be attenuated through washout filters to confine the motion-based flight simulator within limited space presents a critical fidelity issue in both flight research and training.

Several studies[1-4] have been done at NASA Ames Research Center to investigate the cueing fidelity requirements for hovering aircraft. A series investigated the motion fidelity requirements for a roll-lateral precision hover task. The experiments studied the roll and lateral motion fidelity requirements based on the coordinated lateral motion component due to roll motion. This is of particular importance to most motion-based flight simulators that operate in multiple degrees-of-freedom (DOF).

Reference 1 and 2 found that visual cues, roll motion cues, and lateral motion cues interactively affect the motion fidelity, and concluded with a set of characteristic criteria on the performance regarding these primary simulation cues. Reference 2 showed how synchronous the roll and lateral cues must be. Reference 3 found that there is a close

[*] Member of AIAA.

[†] Senior Member of AIAA.

relationship between the motion fidelity and the handling qualities.

Reference 4 has found that coordinated lateral motion is important to the overall motion fidelity in a sidestep task. That conclusion suggests for motion-based flight simulations, rotational motions have to be accompanied with proper coordinated translational motion. Otherwise, the uncoordinated specific force due to angular motion could generate adverse motion perception and influence the motion fidelity with respect to perceived visual cues. This result has profound significance to motion-based flight simulator operators. The coordinated lateral motion commands are independent of the specific forces applied and are usually generated by cross-feeding the aircraft roll response through a translational washout filter, Figure 1, to restrain the motion within the limited translational travel. That is, the trade-off has to be made for motion-based flight simulators.

Reference 4 has also found that there is a significant relationship between the magnitude of the roll motion and the amount of coordinated lateral motion and proposed a combined motion fidelity criteria. Another conclusion from Reference 4 is that , even for tasks with less than 0.1 g's of acceleration[5], coordinated fidelity criteria are required.

Since the motion commands are generated by the washout filters in almost all motion-based flight simulators, two frequency dependent aspects of these filters, i.e., magnitude and phase, warrant careful investigation. Reference 4 only investigated the magnitude effect in a sidestep task without using washout filters. The main objectives of this study were therefore to investigate if the overall motion fidelity depended on coordinated lateral washout filters, and to developed a more general criteria for motion-based flight simulator operators.

Experiment Description

Coordinated roll-lateral motion

With a typical fully coordinated roll-lateral motion, where body-axis lateral aero-propulsive forces are zero, the pilot will not sense any side force providing that the pilot is at the rotational center. However, if the coordinated roll-lateral motion is not treated properly in ground-based flight simulator, or is being compromised to preserve the lateral travel, the immediate effect is that the erroneous specific force will be introduced to the pilot as shown in Figure 2.

Aircraft Model, Force Characteristics, and The Task

A two DOF helicopter model with a roll rate command and a fully coordinated roll-lateral response about hover is given by equation 1,

The roll acceleration due to roll rate (or the roll damping stability derivative), L_p, was set at -4.5 1/sec and the roll control power, $L_{\delta_{lat}}$, was 1.6 rad/sec^2/in. The control sensitivity was selected to have the similar roll rate per stick force response as References 3 and 4. The rotational center was set at the pilot's abdomen. The lateral stick force feel characteristics are shown in Figure 3 with a 1 lb breakout force. The force gradient characteristics came close to Level 1 handling qualities specification of 2.5 lb/in according to the *Handling Qualities Requirements for Military Rotorcraft*[6]. An experienced test pilot, who did not participate the test, flew the task and determined the response of the helicopter was satisfactory.

The Task

The task was a 20 ft sidestep task performed at a constant altitude of 25 ft as shown in Figure 4. The piloted task started with a 20 ft translation towards the desired hover position to the right followed by 10 seconds of stationkeeping. This was followed by a 20 ft translation back to the initial hover position and another 10 sec stationkeeping at that position. Each stationkeeping point was denoted by the center of the two red circles, and a noseboom was placed on the aircraft to allow for alignment with the circle's center. The black and white striped visual layout with two designated red hover circles were reproduced with the same dimensions as Reference 4. Pilots were instructed to complete the right and left sidesteps in one smooth maneuver within a specified time. The desired time to complete the lateral translation was 6 seconds. The adequate completion time was 10 seconds. The desired stationkeeping position error was ±2 ft which was matched by the diameter of the two red circles for easy identification. The adequate position error was ±5 ft.

Three experienced pilots participated in this investigation. Pilots were asked to give handling qualities ratings (HQRs)[7] and motion fidelity scale (MFS) ratings as shown in Table 1.

Test Facilities

The Lockheed Martin Skunk Works Research & Development Simulator (LMSWRDS) is a six-DOF hexapod motion platform. The LMSWRDS system includes a CAE six degree-of-freedom, six actuators (80 in extension), Model 750 motion system. Its roll and lateral motion capabilities are shown in Table 2. The host real-time computer was a Gould 9780, which was running with a frame rate of 60 Hz. The cockpit has a field-of-view (FOV) of 139 degrees of azimuth and 26 degrees of instantaneous elevation which also closely matches the FOV used in Reference 4. The simulator includes a visual system capable of very low transport delay (25 milliseconds) measured from host computer output to when the three channel visual display is completely updated. An additional 33 msec was added to the visual command to make a total theoretical visual throughput delay of 58 msec to closely match that used in Reference 3, which used the same aircraft model and the task to investigate the motion fidelity effects on handling qualities.

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} L_p & 0 & 0 \\ 0 & 0 & g \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ \phi \end{bmatrix} + \begin{bmatrix} L_{\delta_{lat}} \\ 0 \\ 0 \end{bmatrix} \delta_{lat} \qquad (1)$$

The actual LMSWRDS visual throughput delay was measured by using a NASA Ames developed Image Dynamics Measuring System (IDMS), which tracked the movement of the video contrast, in a setup as shown in Figure 5. The throughput delay measured from the model attitude command to recorded video feedback with a white noise clearly shows a frequency response of a pure time-delay of approximately 70 msec, Figure 6. The difference of approximately 12 msec between the theoretical delay and measured delay can be attributed to half of frame-time (about 9 msec) due to digital to analog conversion of the IDMS signal and the real-time operating system overhead.

A four-axis accelerometer package (roll, longitudinal, lateral, and heave in body-axis) and a three-axis angular rate gyro package were installed in the motion compartment near the pilot's headrest. An inverse-transformation algorithm[8], which converts the hexapod's actuator positions to cab attitudes and translational travels, was integrated to supplement the other instrumentation for all simulator motion states in the roll and lateral axes. These motion system responses were used to calibrate the roll and lateral motion cues, to characterize LMSWRDS dynamic characteristics, and to monitor the motion system responses during the experiment.

Motion Cues Response

The motion system responses were checked using the frequency response technique developed for system identification called CIFER®[9]. The test procedure[3] by applying the white noise with a Gaussian distribution was used as the motion command input in the LMSWRDS. LMSWRDS frequency responses of the roll and lateral motion, Figure 6, exhibited good coherence up to 20 rad/sec, which was beyond the frequency range of the pilot input for the task. Both roll and lateral responses show good agreement with each other and generally follow the visual throughput delay response.

Test Configurations

Two washout filters, Figure 1, were developed for this investigation. A roll washout filter generated the roll motion commands, and a coordinated lateral washout filter provided the coordinated lateral motion due to roll motion. Three roll washout filter characteristics, Figure 7, were selected to represent low, medium, and high motion fidelity as defined in Reference 10 for the angular motion fidelity. The three roll filters were effectively first order washout filters though they had the second order form. These test points were also consistent with Reference 3. Reference 4 suggests a gain criterion for coordinated lateral motion due to roll, Figure 8. Four coordinated lateral washout filters with variations in phase characteristics only, as shown in Figure 9, were developed to investigate the phase effect of the sidestep task. The gains for these four coordinated lateral washout filters at high frequency were all unity. The gain and washout filter frequencies used for the three roll configurations and four coordinated lateral configurations are shown in Table 3 and 4 respectively.

Preliminary Results

A representative time history of the sidestep task is shown in Figure 10. The time history shows the first and second phases of the task when the pilot translated 20 ft to the right and maintained stationkeeping for 10 sec. Time traces for four different coordinated washout filters with the medium roll fidelity configuration are shown. Time trace of a closer look at the helicopter's roll attitude and a_y for four coordinated washout filters are shown in Figure 11. It clearly shows that measured lateral accelerations, or uncoordinated specific forces, increase as the fidelity of the coordinated lateral motion degrades.

The averaged Motion Fidelity Scale (MFS) ratings are shown in Figure 12. The averaged HQRs are shown in Figure 13. In general, the low and medium roll motion fidelity configuration data show the expected response from pilots in both fidelity perception and expected handling qualities. There is an apparent effect due to the change of the roll motion fidelity. At medium roll fidelity with full coordinated lateral motion, the test data show good match with the data from Reference 3 which were taken from a different motion platform with the exact same model characteristics and the task.

The LMSWDS ran out of lateral travel with the two coordinated lateral motion configurations that had better phase characteristics under high roll fidelity. They did not receive the expected subjective response from the pilots. That was the obvious reason why the data shown in Figures 12 and 13 differed from Reference 3. Also, it should be noted that in Reference 3 it was the noisy translational track given by the pilots as the primary reason to downgrade the MFS rating from high to medium. Two of the lesser fidelity configurations under high roll fidelity, i.e., higher relative phase difference between the roll and coordinated lateral motion (at 43° and 80°), were operated within LMSWRDS allowable travel and received poor MFS and HQR ratings. This shows that the phase lead due to the coordinated lateral washout filter has a negative effect to the overall motion fidelity. The results also suggest that high roll motion fidelity is not achievable with the unity coordinated washout gain for this task. This is consistent with the findings from Reference 5.

With the low roll motion fidelity configuration, increasing the phase lead of the coordinated lateral washout filter with respect to the roll motion, did not have an apparent effect on averaged MFS nor HQR ratings. Most pilots felt the motion fidelity was objectionable, including the fully coordinated case. However, in the medium motion fidelity configuration, averaged MFS and HQR ratings degraded when the relative phase lead between the roll and coordinated motion was increased to 80 degrees which again showed the phase effect due to the coordinated lateral washout filter.

By combining the gain effect[4] and phase effect due to the coordinated lateral washout filter, a possible criterion shown in Figure 14. This criterion is similar to the motion fidelity criteria in Reference 10. However, the results also show there is a strong tie between the roll motion fidelity and the coordinated lateral motion washout filter phase characteristics on the overall motion fidelity. That is consistent with findings from Reference 4. A more general criterion is proposed to determine the overall motion fidelity as shown in Figure 15. This criterion is also consistent with all the findings from Reference 1 through 4.

Concluding Remarks

1) Pilot's confidence on motion fidelity is dependent on the combined effect of roll and lateral motion cues. Therefore, the criteria as suggested in Reference 4 is an appropriate form to show the dependency between the roll motion and the coordinated lateral motion.

2) Based on the results of this investigation, a general motion fidelity criterion regarding the roll motion gain, the coordinated lateral motion gain and the relative phase between the two motion washout filters is proposed.

References

[1]Chung, W.Y. and Schroeder, J.A.: "Visual and Roll-Lateral Motion Cueing Synchronization Requirements for Motion-Based Flight Simulations," American Helicopter Society's 53rd Annual Forum, April 1997.

[2]Chung, W.Y., Schroeder, J.A., and Johnson, W.W.: "Effects of Vehicle Bandwidth and Visual Spatial-Frequency on Simulation Cueing Synchronization Requirements," AIAA Atmospheric Flight Mechanics Conference, AIAA-97-3655, August 1997.

[3]Chung, W.Y., Schroeder, J.A., and Robinson, D.: "An Initial Evaluation of the Effects of Motion Platform and Drive Characteristics," AIAA Motion and Simulation Techonologies Conference, AIAA-97-3503, August 1997.

[4]Schroeder, J.A. and Chung, W.Y.: "Effects of Roll and Lateral Flight Simulation Motion Gains on a Sidestep Task," American Helicopter Society's 53rd Annual Forum, April 1997.

[5]Jex, H.R., Jewell, W.F., Magdaleno, R.E., and Junker, A.M.: "Effects of Various Lateral-Beam Washouts on Pilot Tracking and Opionio in the Lamar Simulator," AFFDL-TR-79-3134, pp. 244-266.

[6]Aeronautical Design Standard, Handling Qualities Requirements for Military Rotorcraft, ADS-33D, July 1994.

[7]Cooper, G. E., and Harper, R. P., Jr.: "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities," NASA TN D-5153, April 1969.

[8]Dieudonne, J.E., Parrish, R.V., and Bardusch, R.E.: "An Actuator Extension Transformation for a Motion Simulator And An Inverse Transformation Applying Newton–Raphson's Method," NASA TN D-7067, December 1972..

[9]Tischler, M. B., Cauffman, M.G.: "Frequency-Response Method for Rotorcraft System Identification: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics," Journal of the American Helicopter Society, Vol 37, No 3, pgs 3-17, July 1992.

[10]Sinacori, J. B.: "The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility," NASA CR 152066, September 1977.

Table 1. Motion fidelity scale

	Description	Score
High Fidelity	Motion sensations are not noticeably different from those of visual flight	1
Medium Fidelity	Motion sensations are noticeably different from those of visual flight, but not objectionable	2
Low Fidelity	Motion sensations are noticeably different from those of visual flight and objectionable	3

Table 2. Roll and lateral motion limits for Lockheed Martin Skunk Works Research and Development Simulator (LMSWRDS)

Axis	Limits	LMSWRDS
Roll	Displacement (deg)	±26.0*
	Rate (deg/sec)	±25.0*
	Acceleration (deg/sec^2)	±160.0*
Lateral	Displacement (ft)	±5*
	Velocity (ft/sec)	±2.67*
	Acceleration (ft/sec^2)	±19.0*

* Maximum limit in a single axis excursion

Table 3. Roll washout filter configurations

Roll Washout Filter Configurations	K_p	K_{ps}	ζ_p	ω_p	Gain	Phase Distortion (deg)
					At 1 rad/sec	
High fidelity	0.45	0.5	1.0	0.5	0.4	25
Medium fidelity	0.25	0.5	1.0	0.5	0.22	25
Low fidelity	0.5	1.0	1.263	1.414	0.2	60

Table 4. Coordinated lateral washout filter configurations

Coordinated Lateral Washout Filter Configurations	K_{lat}	ζ_y	ω_y	Gain	Phase Distortion (deg)
				At 1 rad/sec	
Fully coordinated (fc), 0° lead	1	0.7	0.01	1	0
20° lead	1	0.7	0.25	1	20
43° lead	1	0.7	0.52	0.98	43
80° lead	1	0.7	0.8	0.8	80

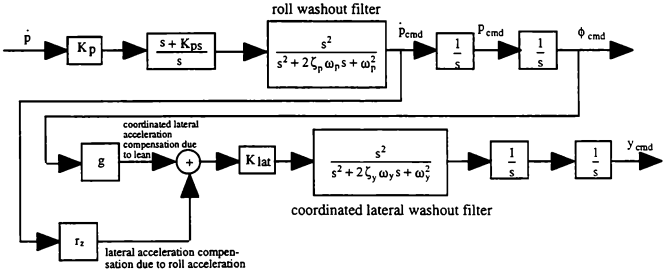

roll washout filter
$\dot{p}$
K_p
$\frac{s+K_{ps}}{s}$
$\frac{s^2}{s^2+2\zeta_p\omega_p s+\omega_p^2}$
$\dot{p}_{cmd}$
$\frac{1}{s}$
p_{cmd}
$\frac{1}{s}$
ϕ_{cmd}
coordinated lateral acceleration compensation due to lean
g
+
K_{lat}
$\frac{s^2}{s^2+2\zeta_y\omega_y s+\omega_y^2}$
$\frac{1}{s}$
$\frac{1}{s}$
y_{cmd}
coordinated lateral washout filter
r_z
lateral acceleration compensation due to roll acceleration

Figure 1. A representative motion drive command block diagram for roll and lateral drives.

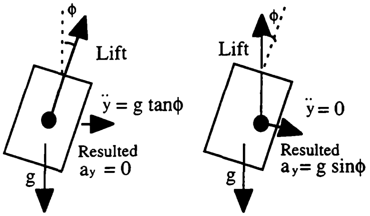


Figure 2. Coordinated vs. uncoordinated Flight

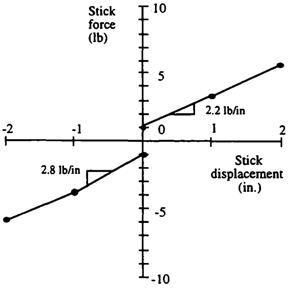


Figure 3. The lateral sitck force gradient characteristics for the Lockheed Martin Skunk Works Research & Development Simulator (LMSWRDS)

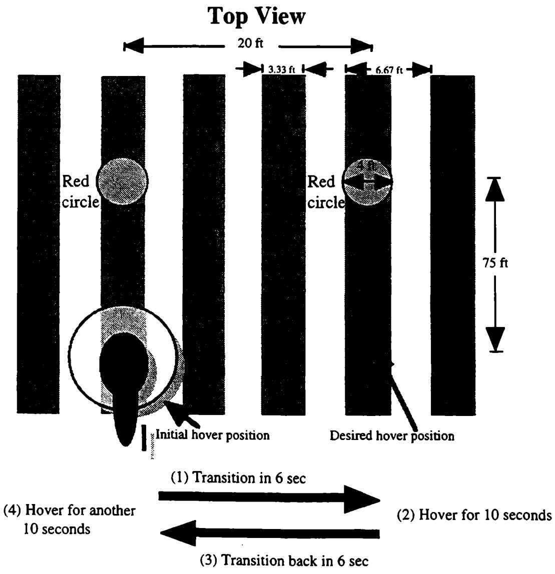


Figure 4. Top view of the task layout

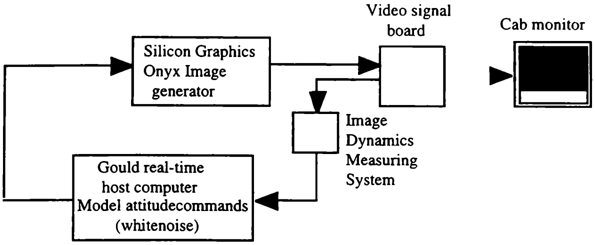

Video signal board
Cab monitor
Silicon Graphics Onyx Image generator
Image Dynamics Measuring System
Gould real-time host computer Model attitudecommands (whitenoise)

Figure 5. Visual delay measurement setup by white noise or frequency sweep

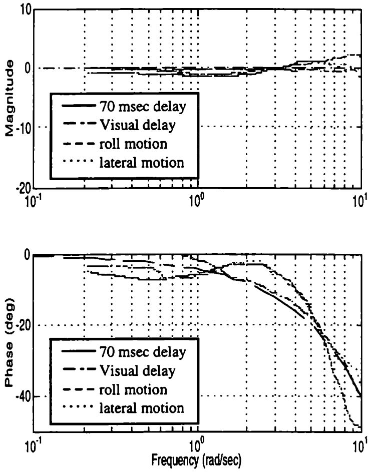


Figure 6. Frequency response of LMSWRDS visual delay, roll motion, and lateral motion.

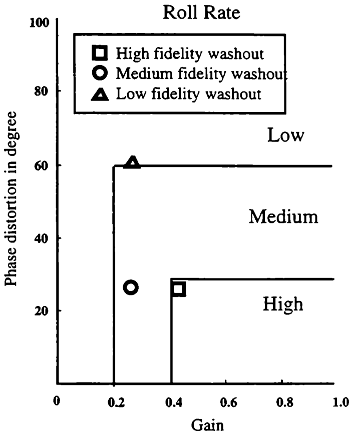


Figure 7 Roll motion fidelity versus phase distortion and gain at 1 rad/sec for the three roll washout motion configurations

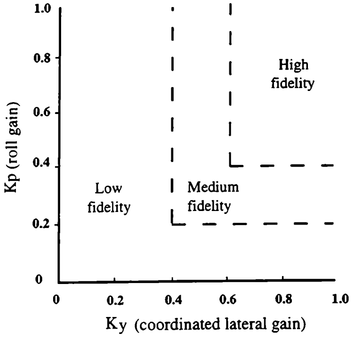


Figure 8. A specification on roll and lateral gain for coordinated roll-lateral motion

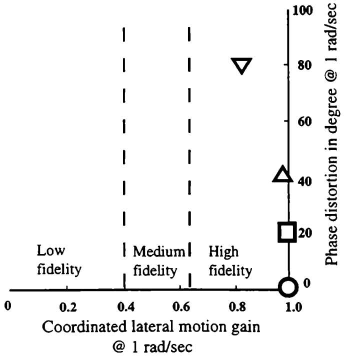


Figure 9. Selection of test points to investigate the phase effect for the coordinated lateral washout filter

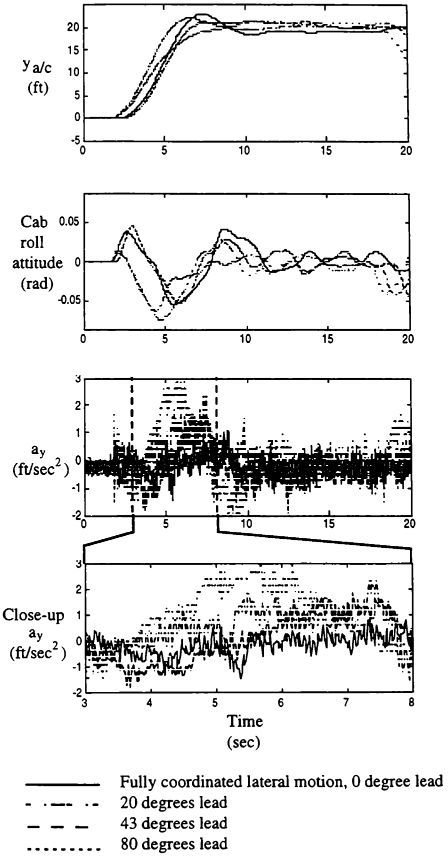


Figure 10. Time history of the four different coordinated lateral washout filter configurations with the medium roll fidelity

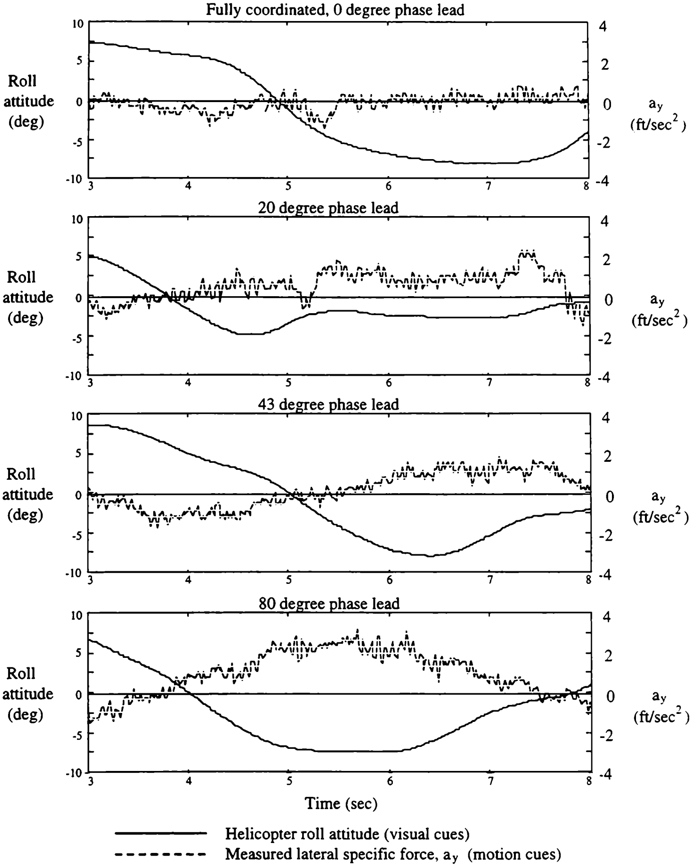


Figure 11. Helicopter roll attitude vs sensed lateral specific force, a_y

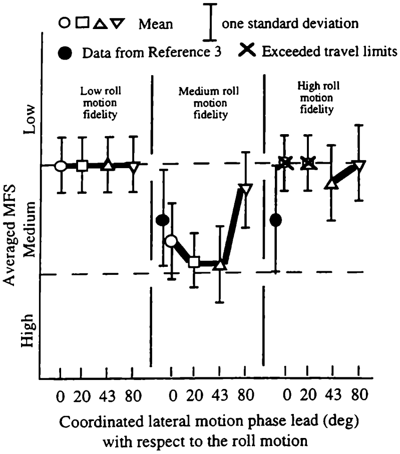


Figure 12. Averaged Motion Fidelity Scale (MFS) with the low, medium, and high roll motion fidelity

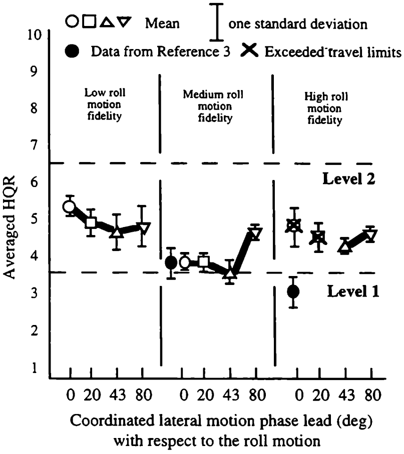


Figure 13. Averaged HQRs with the low, medium, and high roll motion fidelity

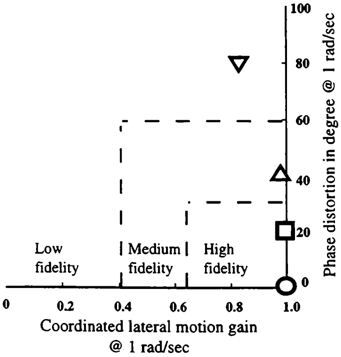


Figure 14. Proposed motion fifdelity criteria for coordinated lateral motion

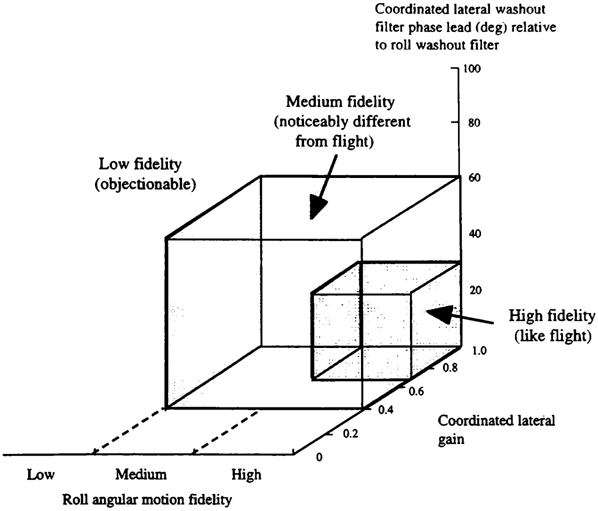

Coordinated lateral washout filter phase lead (deg) relative to roll washout filter
100
80
60
40
20
Medium fidelity (noticeably different from flight)
Low fidelity (objectionable)
High fidelity (like flight)
1.0
0.8
0.6
0.4
0.2
0
Coordinated lateral gain
Low
Medium
High
Roll angular motion fidelity

Figure 15. Proposed overall roll motion fidelity criteria based on the gain and phase distortion at 1 rad/sec

AIAA-98-4173

THE BLUR ZONE – CUE OR SENSATION?

A G Barnes[*]
Consultant
Lytham St Annes
UK

Abstract

Clarity of vision is reduced when a rapidly moving object is viewed form a short distance. A familiar illustration is our inability to read the name of a railway station from a high-speed train, because the image is blurred. Similarly, the image of the ground below a fast moving vehicle is blurred. The transition from clear vision to blurred vision of the passing scene is not a gradual process; blurring quickly occurs when a threshold value of the rate of movement of the image relative to the eye is exceeded. The result is an area in the observer's field of view referred to as the "blur zone". The size of the blur zone depends on speed, and the proximity of the viewed object.

The ground seen from an aircraft will only be blurred at relatively low height/ high speed. A 1965 Agard paper, addressing the problems of low level military flying, suggests that a pilot's ability to detect and recognise targets could be impaired by the blur zone. The possibility that the blur zone could in some circumstances help the pilot has not been generally considered. Much research has been done on the visual cues used in low flying, take off, and landing - orientation, perspective, size of objects, ground texture, and so on. This paper adds the blur zone to the list, and examines its significance.

The nature and shape of the blur zone are defined, and related to the view from the cockpit of different aircraft, in circumstances where the blur zone may be visible. The question arises whether pilots subconsciously use the blur zone as a feedback, to improve performance, or whether it is simply enhances the impression of being close to the ground. For example, when landing a small aircraft on grass, blurring may help in judging the last few feet. A crop sprayer may "fly" the blur zone. Pilots of civil transport aircraft will not see it, except in ground roll prior to take-off, and it will only provide a sensation of speed.

There is room for further investigation. In particular, existing flight simulators may not reproduce the effect of the blur zone, because of their image generation or hardware limitations. The effect could be represented artificially, possibly by darkening the affected area in low cost visual ·· systems. Only a small part of the flight envelope benefits, but it is an important area, where additional realism is needed in simulators.

Contents

1. Introduction

The subject of pilot cueing has been of interest to researchers since the earliest days of flight. An understanding of how the pilot uses information to maintain control of his aircraft, and to perform specific tasks, leads to improvements in aircraft design, cockpit layout, and operational safety.

In particular, the nature of the feed-backs used by the pilot for closed-loop control determines the layout of the primary flight controls. It is self-evident that visual cues dominate, and that control of aircraft attitude requires knowledge of angular orientation. Outer-loop closures, such as heading, sink-rate, and height, are also made from visual information. Perspective plays a part in the determination of position relative to other objects, and proximity to the ground enhances the estimation of speed.

Much has been written on the subject of visual cueing, and how mis-interpretation of cues can occur in special circumstances, leading to reduced performance, or even pilot dis-orientation. One problem associated with flying military aircraft at high speed and low level is the difficulty of detecting

[*] Senior Member, AIAA

and recognising targets on the ground, as vision is impaired when the target enters the 'blur zone'.[1] The blur zone is a volume of space around a moving observer, within which static objects become blurred. It is associated with the inability of the eye to resolve objects with a high sight-line spin rate. The critical spin rate used in Reference 1, above which blurring occurs, is 100 degrees/second. The size of the blur zone depends on speed and the proximity of viewed objects.

Many years ago, an ex-wartime flying instructor told the author how to make a perfect landing in a Tiger Moth (a small bi-plane which, because it has a tail skid, can only operate from grass fields): "delay the flare until you see individual blades of grass". A few years later, another veteran pilot reversed this advice. His way was to "start the flare when you can't see the blades of grass".

Recently, while preparing a paper on pilot cueing, the thought occurred that there could be a link between these contradictory remarks and the blur zone. The blur zone may be a useful indication to pilots of ground proximity, and may subconsciously assist them when manoeuvring close to the ground.

The purpose of this paper is to explore this possibility. No mention of the blur zone is made in the usual references on pilot cueing, and pilots do not cite it as a factor. In most conditions of flight, external objects are not close enough to be in the blur zone. Nevertheless, the blur zone does exist, and if an aircraft in flight is sufficiently close to the ground, it will contribute to the sensation of speed. By defining the shape and size of the blur zone, and relating it to realistic flight situations, this paper questions if its value is restricted to being only a sensation, or whether it occasionally contributes to, and affects control strategy.

2. Nature and Size of the blur zone

The appearance of the blur zone may be judged by viewing the textured surface of the road, ahead of a moving car. Far ahead, no blurring occurs, but as the driver's sight line is lowered, a depression angle is reached, below which blurring occurs. The boundary between the blurred and non-blurred image is rather indistinct, moves with the car, and moves towards the horizon as the speed of the car increases. The clarity of the boundary depends on factors such as the textural detail, lighting conditions, and the visual acuity of the observer. Even so, it is readily observed, and extends all around the car.

Over flat terrain, the blur zone is an area on the ground, centred below the eye-point, formed by the intersection of the ground plane and a torus, or doughnut shape surrounding the pilot (Figure 1). The size of the torus is proportional to forward speed, and inversely proportional to critical sight-line spin rate. The area of the intersection is dependent on the ratio of height to velocity, h/V. The derivation of the blur zone is given in the Appendix.

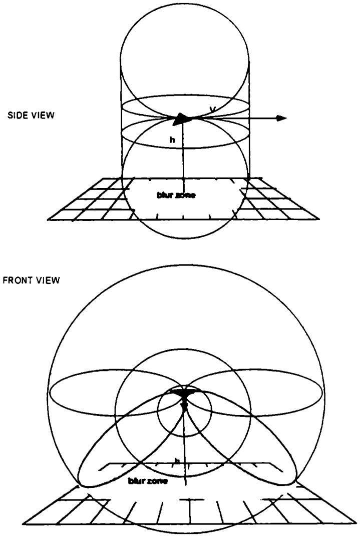


Figure 1. The blur zone - the intersection of the ground plane and a torus formed by rotating a circle about the eye point.

For an aircraft flying at 600 knots, close to the ground, the blur zone will only be seen by an observer in the aircraft looking down, if the eye-height is less than 600 feet. The dimensions of the blur zone increase rapidly with decreasing height, and it will not normally be discernible ahead of the aircraft, due to vision restriction caused by the cockpit structure. As height decreases, it will be seen peripherally, or by looking sideways. A typical cockpit view from 100 feet is given on figure 2. The size of the blur zone is proportional to speed over the ground - as speed increases, the blur zone moves closer to the horizon.

The higher sight-line spin towards the centre of the blur zone increases the blurring, and gives a greater impression of speed over the ground. Both the position of the edge of the blur zone, and the appearance of the ground within the zone, may contribute to a pilot's impression of how low he is flying.

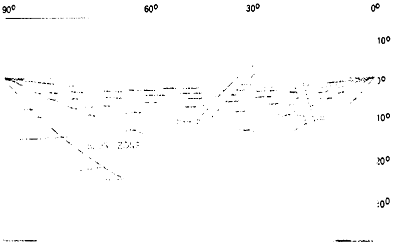


Figure 2 Field of view (Tornado) - blur zone, high speed low level

3. The Blur Zone as a Piloting Cue

The blur zone has not been formally investigated as a feedback cue to assist in controlling aircraft at very low height. It is likely that pilots are subconsciously aware of its presence. With experience, it is possible that they can use it to supplement more established parameters

- attitude
- velocity vector (determined from stream-lines)
- rates
- accelerations
- perspective
- objects, such as trees and buildings.

It is well recognised that these parameters are used, to a lesser or greater degree according to pilot technique and prevailing conditions, and that there is redundancy of information.

Three flight regimes where the blur zone may assist the pilot's control task are discussed below.

3.1 High speed, low level flight

The blur zone can only be seen from the cockpit, and is only significant for small values of h/V. Figure 3 shows the blur zone for flight at 50 feet and 600 knots (or 20 feet and 240 knots). Such flying may be regarded as skilled, or even reckless, but does occur in circumstances such as wartime attacks, air displays, or low flying over the sea. The blur zone is most compelling over flat, featureless (but textured) surfaces, such as desert or sea, and will be increasingly compelling at very low height, particularly in turns. Moreover, if the speed is known, absolute height (rather than relative height) can be inferred.

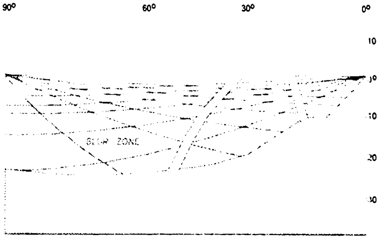


Figure 3 Field of view (Tornado) - blur zone, 600 knots, 50 feet

For specialist operations, which call for precision flight at low level, such as dropping stores and supplies, or crop spraying, the blur zone could well be a vital cue. It is possible also that helicopter pilots in forward flight, operating at low level, are aware of the blur zone, because of the larger downward field of view compared to fixed wing aircraft.

3.2 Landing

Landing techniques vary with the type of approach (curved or straight-in), and with the type of aircraft.[2] The skill in landing a large jet is required at an early stage in the approach. Large aircraft are less responsive to high frequency control inputs, ground effect is usually favourable, and turbulence is less disturbing. The size of the aircraft dictates that on touchdown, the pilot's viewpoint is high, and so it is unlikely that the blur zone plays any part in control strategy. The landings are made on prepared pavements of known dimensions, with helpful markings, and aids to assist in setting an accurate approach and touchdown point. Figure 4, the blur zone for a Boeing 747 at touchdown, shows that the pilot is unlikely to even see it.

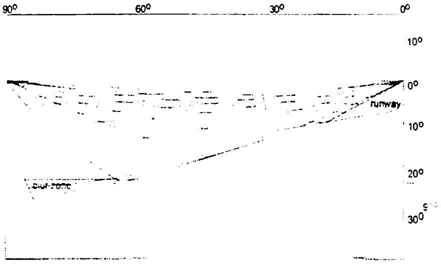


Figure 4. Boeing 747 at touchdown - 120 knots - 8° alpha - eye height 40 feet

More active control participation is required by the pilot of a small aircraft during the approach, flare, and touchdown, because it is more sensitive to control inputs, and to turbulence. There is a need to 'feel' for the ground. Even more attention is needed to land a tail-sitter than to land an aircraft with a tricycle gear. The task of landing a Tiger Moth (or a Spitfire) on a grass airfield is in sharp contrast to the landing of a large jet transport on a runway. The lack of forward visibility requires the pilot to use off-centre visual cues, sometimes combined with a curved approach down to the flare. During the flare, there are no runway markings, perspective cueing is poor, and dependence is placed on the textural appearance of the grass. Although the speed is low, the ratio of h/V is also low at touchdown, the blur zone will certainly be visible to the pilot, and may be a useful cue to start the flare.

The restricted view ahead in single engined aircraft requires the pilot to use the view sideways. Immediately prior to touch-down, inexperienced pilots are often not sure what they should do to ensure a soft landing. Ground contact (or lack of it) may come as a surprise, and a good landing is often credited to good fortune. More experienced pilots are better able to judge height in the round out. Two field of view plots for the Cessna 150 (a high wing light aircraft) are shown on Figure 5. At a ground speed of 60 knots, the changing appearance of the blur zone in the last few feet prior to touchdown illustrates that it may well be a useful cue.

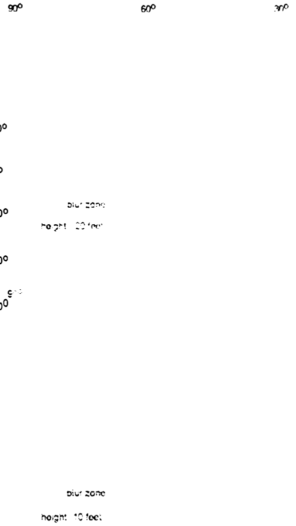


Figure 5. Field of View Plots Cessna 150M 60 knots 20 feet and 10 feet

3.3 Ground Roll/Take-Off

The take-off task has a strong open-loop content. Full power is applied, speed is monitored, directional control is maintained, and pitch control is applied at the appropriate point. Watch is maintained for any digression from the norm, which requires attention inside and outside the cockpit. Knowledge of speed is important throughout, for reasons both of performance and safety, and outside visual cues can greatly reduce the need to read the airspeed indicator. It is well known from flight simulator experience (and from take-offs at night) that if no information comes from the texture of the runway surface, speed is under-estimated.

The sensation of speed triggers the actions of nose-wheel lifting, and un-stick. The blur zone is detectable from most cockpits during take-off. As the edge of the blur zone moves progressively towards the horizon with increasing speed, it could be used as confirmation that the point for these actions has been reached. In circumstances where out-of-cockpit view must be kept, such as a formation take-off, this form of speed monitoring may well ease the pilot's task.

4. Significance of the blur zone

Whether or not the blur zone contributes to the above tasks first depends on how easily it can be detected, and how reliable the information is. The underlying mechanism is tied to the complexities of image perception, and how visual information is processed. Images we see in the cinema or on TV illustrate the ability of the brain to integrate sequential pictures to give continuity.

More than sixty years ago, it was shown that the human visual system summates signals over time (about 120 ms in daylight), to enhance visual sensitivity. Rapidly moving objects smear. More recent work has shown that if a moving target is exposed long enough to elicit a clear sensation of motion, the amount of smear is less than would be expected, by comparison with photographs taken with a 120 ms shutter speed.[3] Only very rapid image motion creates smear.

The above work suggests that the appearance of the edge of the blur zone will be distinct, as the compensation mechanism breaks down. The measurements were made using a static observer in a laboratory. The dynamic and stressful conditions of flight at low level may well change the boundary onset. It is likely also that the textural quality of the scene (regularity, shape, and contrast) has an influence. As state earlier, the appearance of the interior of the zone is important. Although features cannot be resolved, the direction of movement, and the impression of speed are evident, and become more compelling with greater immersion in the blur zone.

4.1 Flying Instruction

Like most motor skills, pilot instruction includes the teaching of theory and principles, successive demonstration of technique, and individual practice. Different tasks and operating conditions call for different techniques, and a redundancy of available information means that even in the same circumstances, skilled pilots may achieve equivalent performance using differing control strategies. Evidence is seen in recordings of control activity of different pilots, which can vary greatly in amplitude and bandwidth.

For good performance, however, pilots develop the ability to detect small deviations from the norm (small error thresholds), and subconsciously respond to stimuli. The blur zone may fall into the category of a sub-conscious cue. It cannot be separated from other cues (a textured surface also provides perspective; translation across it adds motion parallax). In most of the flight envelope, there is no blur zone to be seen, and the question this paper raises is whether it is useful to a pilot, when it does appear.

4.2 Flight Simulation

The real significance of the blur zone may lie in the field of flight simulation. Flight simulators are still deficient in representing both flight at low level, and the landing phase of flight - precisely the areas discussed in Section 3. The deficiencies are associated with the shortcomings of current visual systems, which cannot approach the capabilities of the eye. Compromises are made between field of view, resolution, contrast, brightness, scene content, and refresh rate of the display.[4] Although blurring of the visual scene occurs in simulations of low level flight, the mechanism is usually associated with the limited performance of the display, rather than the eye.

The art of simulation, however, is to overcome known deficiencies in technology by combining the available performance for best results. For example, setting brightness and contrast levels of the display, and using exaggerated colours, to match or compensate for the poor resolution. It would also be possible to depict in the visual display, an artificial blur zone, as seen on figures 2 and 3, as a darkened area, below, and moving with the aircraft, changing with height and speed. A sharp edge to the boundary, and a progressive darkening, could be easily provided. Representing the dynamic quality of the blur is more of a challenge, but is feasible

.4.3 Cue or Sensation?

The blur zone may or may not be a useful cue to the pilot, to enhance performance in special circumstances, as described in Section 3. Pilots are interested to discuss the possibility, and develop ideas. For example, accidents have occurred in which aircraft inadvertently hit the sea, in conditions when the sea is unexpectedly calm (no texture), and haze obscures the horizon. The lack of blurring could be a significant factor in such mis-judgements.

However, all pilots (and anyone in a moving vehicle) perceive the blurring of static objects and the terrain, more noticeable with increasing speed and proximity. Combined with other sensations, such as noise and vibration, it contributes to the sensation of speed.

In many areas of flight simulation, it is important to strive for realism.[5] Pilots are more likely to behave in the simulator as they do in the aircraft, if the simulator is realistic. Even if the blur zone is a sensation, rather than a cue, its mechanism, as described in this paper, should be taken into account by outside-world visual display designers. A ground based flight simulator is the ideal tool to investigate the blur zone.

5. Conclusion

Many words have been written on the visual cues used by pilots to control an aircraft in different phases of flight. This paper identifies one cue that has not attracted much formal study, possibly because it cannot be observed in most phases of flight. The blur zone will only be seen under extreme combinations of high speed and/or low altitude. However, these conditions, - low flying, landing and take off - require skilled flying, and can be hazardous.

The appearance of the blur zone depends on the way in which the eye and brain combine to create visual impressions, and the mechanisms are briefly presented. The appearance of the blur zone is a common experience, but the extent to which pilots can and do use the information it contains is uncertain.

Further work on the topic would be justified. In particular, ground based simulators could be used to see whether a stylised blur zone is helpful. If so, there is a prospect of improving the effectiveness of simulators in specific circumstances.

6. References

6.1 AGARDograph 107 "Problems of Vision in Low Level Flight" Prof. A Mercier, Lt. Col. G Perdriel, Wg. Cdr. T C D Whiteside AGARD Aerospace Medical Panel 1965

6.2 AGARD Conference Proceedings CP 249 "Simulating the Visual Approach and Landing" A G Barnes FMP Symposium on "Piloted Aircraft Environment Simulation Techniques" held in Brussels April 1978

6.3 "Nature" Vol. 284, pp 164-165 "Motion Smear" David Burr Dept. of Experimental Psychology, Cambridge 13 March 1980

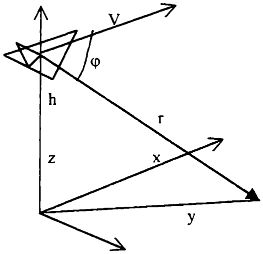
6.4 AGARD Advisory Report AR 164 "Characteristics of Flight Simulator Visual Systems" ed. Dr I C Statler May 1981

6.5 AIAAA -91-2920 "The Compromise between Accuracy and Realism in Flight Simulation" A G Barnes Flight Simulation Technologies Conference. New Orleans August 12-14, 1991

7. Appendix

A Formulation of the Blur Zone

The view of an object moving relative to the viewer is modified if the rate of sight-line spin of the object is high. It is suggested in Reference 1 that rapid deterioration of the image occurs if this rate exceeds 100°/second. On this basis, a mathematical expression is derived below to define the area on the ground, which will appear blurred to the pilot of an aircraft flying straight and level, at constant speed, V. Thanks are due to Mr. Mats Lundberg, of Saab AB, Linkoping, Sweden, for this derivation.



Assume an object, fixed in space, at a distance r from an aircraft moving with velocity V at height h, subtending an angle Φ at the pilot's eye. The rate of sight-line spin is

$$\frac{d\phi}{dt} = \frac{V\sin\phi}{r}$$

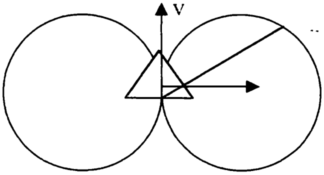
The blur zone is defined by the sight-line spin rate exceeding B =100°/second.

$$B = \frac{V\sin\phi}{r}$$

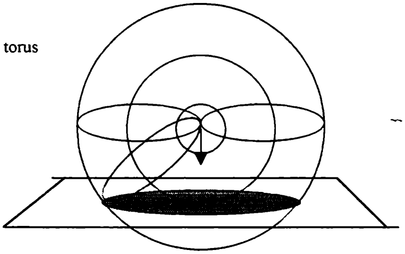
which gives

$$\frac{rB}{V} = |\sin\phi|$$

At zero height (the x-y plane), this equation defines two circles to the left and right of the aircraft, with diameter V/B:



For non-zero heights (three dimensions), there is axial symmetry about the aircraft velocity vector, and the blur zone is created by rotating the circles about the x axis, forming a torus, or doughnut with a zero radius centre. The area on the ground in which objects are blurred is the intersection of the torus with the ground plane. The shape depends on aircraft height, varying from two adjacent circles (zero height) to an ellipse (as h approaches V/B).



In Cartesian co-ordinates, (x, y, h) define a point on the ground.

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\sin\phi = \frac{\sqrt{y^2 + h^2}}{\sqrt{x^2 + y^2 + h^2}}$$

The blur zone is then defined by

$$\frac{B}{V} = \frac{\sqrt{y^2 + h^2}}{x^2 + y^2 + h^2}$$

which gives

$$x^2 = \frac{V}{B}\left(\sqrt{y^2 + h^2}\right) - \left(y^2 + h^2\right)$$

for real x

$$= 0.87V_{knots}\sqrt{y^2 + h^2} - \left(y^2 + h^2\right)$$

for x, y, h in feet

The edge of the blur zone, in angular terms from the pilot's eye point, is given by

Azimuth

$$\sin\psi = \frac{y}{\sqrt{x^2 + y^2}}$$

Elevation

$$\sin\theta = \frac{h}{\sqrt{x^2 + y^2 + z^2}}$$

AIAA-98-4174

STRUCTURED SPECIFICATION OF EXOCENTRIC FRAMES OF REFERENCE

Dr. E. Theunissen,
AIAA member
Delft University of Technology, Faculty of Information Technology and Systems,
P.O. Box 5031, 2600 GA Delft, The Netherlands.
E.theunissen@et.tudelft.nl

Abstract

The frame of reference determines from where a certain (artificial) environment is seen. The most common option for the presentation of navigation data is a 2-D planar display format. With an exocentric frame of reference, range and position error resolution can be selected more independently. It is discussed how requirements in 2-D display space can be translated into a design space which determines the frame of reference and the viewing volume in the 3-D world. Also, an example application is illustrated. Three different frames of reference, including an exocentric one, are compared for their suitability to support a ground movement task in terms of tracking performance, average velocity and conflict detection capability.

Introduction

Based on the frame of reference, spatially integrated data presentation can be classified into egocentric display formats and exocentric display formats. In an egocentric display, the three-dimensional world is depicted as seen from the vehicle under control. In an exocentric display, the situation is viewed from another position. In the area of perspective flightpath displays, an egocentric frame of reference is generally considered superior for guidance and control, whereas an exocentric frame of reference better supports navigation awareness[5]. When translating display concepts for navigation and control to operations on the ground, the pilot would be presented with a flight director for lateral control, a plan view display for airport navigation, and a speed tape for velocity control. An implementation in which a 2-D command display similar to a flight director (FD) was used to present the commanded steering angle and the commanded velocity is discussed by Möhlenkamp and Schänzer[3]. With an FD, the control task is isolated from the navigation task.

The preview presented by the navigation display is not sufficient to allow anticipatory control. Furthermore, when the pilot does not frequently scan the navigation display, it is difficult to distinguish between commands which follow from a change in the direction of the required trajectory and commands which result from position and/or orientation errors. To support the pilot with maintaining a cognitive link between the control task and the navigation task, one can consider presenting guidance requirements instead of control commands. Since the pilot does not have to control aircraft altitude, such a presentation could consist of a track-up plan view depiction of the aircraft and the trajectory constraints. With a plan view display, there is only a single parameter which controls both the visible range and the magnitude of the position error cues. The magnitude of the cues providing position error data is inversely proportional to the visible range. With an exocentric frame of reference (Fig. 1), range and position error resolution can be selected more independently.

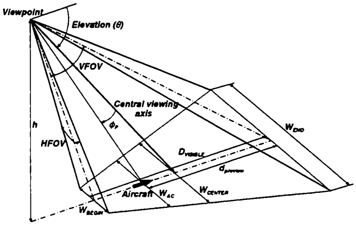


Fig. 1. Example of an exocentric frame of reference. The shape ans size of the visualized area can be controlled through the viewing distance, direction and the field of view.

Fig. 2 shows the visible area presented with an exocentric frame of reference as in Fig. 1, and Fig. 3 the visible area of a conventional plan view display.

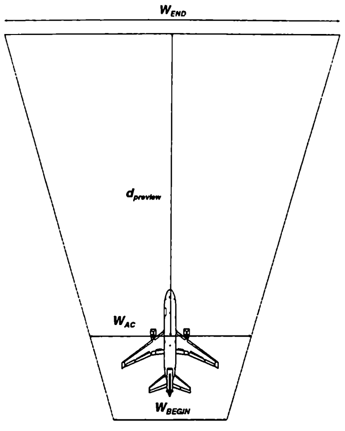


Fig. 2. Planar view of the area presented with an exocentric frame of reference.

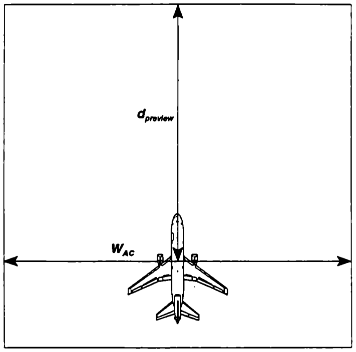


Fig. 3. Plan view display format. The resolution of track deviation is inversely proportional to the display range.

An exocentric display requires the specification of a frame of reference, comprising the viewpoint and the viewing direction, and a geometric field of view.

The viewing direction is specified through azimuth and elevation. With respect to azimuth, Lasswell and Wickens[2] conclude that '*assuming that local guidance is the predominant task involved in taxiing an aircraft, the literature allows for some definite conclusions regarding an appropriate display perspective. The display should be ego-referenced, matching the momentary direction of the aircraft by rotating (e.g. track up) in accord with pilot control inputs to the aircraft. Such a feature will reduce the cognitive demands associated with mental rotation and image comparison. There would also be a benefit of including some degree of world-referenced information for preview of upcoming maneuvers and enhanced awareness of potential hazards*'.

Boyer and Wickens[1] discuss the selection of values for elevation and geometric field of view (GFOV). With respect to the elevation of the viewing direction, they conclude that '*it appears as if both perspective tracking and position judgement is best supported by a 45 degree elevation angle*'. For vertical and forward relative motion, a 45 degree elevation angle in an exocentric frame of reference yields visual cues with approximately the same magnitude. As a result, an elevation of 45 degrees is only optimal in case the task requires equally strong visual cues for both directions of motion. When using an exocentric frame of reference to combine the advantages of a high position error gain with a sufficient amount of trajectory preview, other requirements will determine the selection of the viewpoint elevation.

With respect to the geometric field of view Boyer and Wickens[1] conclude that '*in general it appears as if a smaller GFOV presents an image within which it is easier to determine the relative position of objects in a perspective display. This can be attributed to the fact that the narrow FOV magnifies the scene and provides a better resolution of the distance between objects*'. The fact that with a smaller GFOV the rate at which the resolution decreases along the projected line of sight is also smaller is likely to be the dominant factor.

A general problem is that there is a strong interaction between the effects of changes of the design parameters and the type and magnitude of the resulting visual cues. In a task oriented design, the designer should not need to worry about specifying the values of the design parameters, but about the required type and magnitude of the visual cues which satisfy the task dictated information requirements. The values for the design parameters should follow from visual cue requirements.

For a taxi guidance display, the selection of the design parameters which determine the viewpoint, viewing direction, and geometric field of view influence the type and magnitude of the position error cues, the available preview range, the compression ratio between the size of objects close to the viewpoint and object further away, the gain of the dynamic cues, the perspective distortion, and the likelihood of control blunders. To allow a structured selection of the design parameters and justify design decisions, a method is needed which describes the relation between the previously mentioned aspects.

From requirements to design parameters

When assuming a track aligned display format, the following parameters must still be specified to establish the frame of reference (Fig.1):

1. Elevation of the central viewing axis (elevation).
2. Height of viewpoint (h).
3. Angle ϕ_p specifying the difference between the central display axis and the axis pointing at the aircraft.

Furthermore, the horizontal field of view (HFOV) and the vertical field of view (VFOV) need to be specified. The specification of values for these parameters should be based on requirements with respect to the resolution of the position error data and the desired preview. Since the area in front of the aircraft is more important than the area behind the aircraft, the central viewing axis can be selected to point to a location in front of the aircraft. Many design requirements can be directly related to properties of the resulting 2-D display space, but the basic design parameters are the position and orientation of the viewpoint in 3-D space and the geometric field of view. For a structured design, the type and magnitude of the task related visual cues in 2-D display space must be expressed as a function of the design parameters. A method is needed which allows the position error resolution, preview range, and scene compression requirements to be translated into the position and orientation of the viewing vector and the geometric field of view. The method should also allow for the specification of constraints on the geometric field of view. This raises the question which parameters in 2-D display space are useful and sufficient. As indicated in the introduction, the main reason for selecting an exocentric frame of reference is that this allows more freedom in the selection of the preview range and the position error resolution. These are both requirements which can directly be related to 2-D display space. Two other parameters are the visible ranges at the near and the far distances which are just visible. These distances, together with the preview range determine the total visible area (Fig. 2), which is directly related to the detection range. Fig. 4 presents these four parameters which can be used to derive the location of the viewpoint.

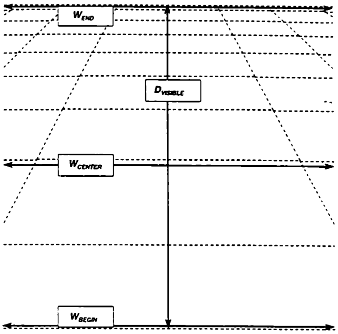


Fig. 4. Design requirements in 2-D display space.

Preferably, the method should allow an independent manipulation of the type and magnitude of the visual cues. The type is determined by the resulting shape of the perspectively presented environment, while the magnitude is determined by the size of the elements in this environment. The following procedure can be used to accomplish this. The position error resolution is determined by the visible lateral range at the position of the aircraft. A position error will result in a displacement of the track relative to the aircraft symbol. When expressing the magnitude of this displacement as a percentage of the total visible range, the position error gain at the center of the display is $100/W_{CENTER}$ [%/m].

When HFOV and VFOV (Fig. 1) are equal to GFOV, the following equations hold:

$$D_{VISIBLE}=\frac{h}{\tan(\theta-GFOV/2)}-\frac{h}{\tan(\theta+GFOV/2)} \quad (1)$$

Furthermore,

$$W_{CENTER}=\frac{-2h\cdot\tan(GFOV/2)}{\sin(\theta)} \quad (2)$$

and

$$\frac{W_{END}}{W_{BEGIN}}=\frac{\sin(\theta+GFOV/2)}{\sin(\theta-GFOV/2)} \quad (3)$$

For a given $D_{VISIBLE}$, W_{CENTER}, and the ratio of W_{END} and W_{BEGIN}, the three remaining unknowns are:

1. elevation θ
2. Geometric field of view GFOV*
3. Height of viewpoint h

Eqs (1) to (3) can be used to calculate the values of these parameters. To get a quick idea of the design options the following method can be used. To allow an independent specification of type and magnitude, rather than using absolute values for these three parameters, the preview ratio $R_{preview}$ of $D_{VISIBLE}$ and W_{CENTER} and the compression ratio R_c of W_{END} and W_{BEGIN} are used to specify the shape of the visible area. In this way, the dependency on the altitude h in the Eqs (1) and (2) can be eliminated, allowing a set of solutions to be generated which yield a particular shape. Now it is possible to calculate the required GFOV and the $R_{preview}$ for a given elevation and R_c. Table 1 provides an overview of $R_{preview}$ for elevations between -5 and -60 degrees, and an R_c between 2 and 9. Table 2 provides an overview of the required GFOV.

Table 1. Preview ratio for different elevations and compression ratios.

	2	3	4	5	6	7	8	9
-5	13	15	18	21	24	26	29	32
-10	6.5	7.6	9.0	10	12	13	14	16
-15	4.4	5.1	6.0	7.0	8.0	8.9	9.7	11
-20	3.3	3.9	4.6	5.3	6.0	6.7	7.3	8.2
-25	2.7	3.2	3.7	4.3	4.8	5.4	5.9	6.6
-30	2.3	2.7	3.1	3.6	4.1	4.6	5.0	5.5
-35	2.0	2.3	2.7	3.1	3.6	4.0	4.4	4.8
-40	1.8	2.1	2.4	2.8	3.2	3.5	3.9	4.3
-45	1.6	1.9	2.2	2.6	2.9	3.2	3.6	3.9
-50	1.5	1.8	2.0	2.4	2.6	X	X	X
-55	1.4	1.6	X	X	X	X	X	X
-60	1.3	X	X	X	X	X	X	X

Since two independent requirements are specified, and the values of three parameters can be controlled to meet these requirements, a design space exists. This space is narrowed down by imposing constraints.

Table 2. GFOV for different elevations and compression ratios.

	2	3	4	5	6	7	8	9
-5	3.3	5.0	6.0	6.8	7.3	7.5	7.8	8.0
-10	6.8	10	12	14	14	15	16	16
-15	10	15	18	20	22	23	24	24
-20	14	21	25	27	29	31	32	33
-25	18	26	31	35	37	39	40	41
-30	22	32	38	42	45	47	48	50
-35	26	39	46	50	53	56	57	59
-40	31	46	54	59	62	64	66	68
-45	37	53	62	68	71	74	76	77
-50	43	62	71	77	80	X	X	X
-55	51	70	X	X	X	X	X	X
-60	60	X	X	X	X	X	X	X

Constraints on the geometric field of view

For small elevations, a change in orientation of the viewpoint will mainly result in a translation of the depicted situation. The rate of this translation is inversely proportional to the geometric field of view. This will impose a lower limit on the geometric field of view. Requirements with respect to minimum angular resolution and maximum allowable perspective distortion[7] pose an upper limit to the geometric field of view.

Examples

Figs 5 to 10 provide examples of three different selections of preview ratios and compression ratios. The dashed lines in these figures represent elevation and azimuth relative to the central viewing axis in increments of 10 degrees. The object which is presented is a grid consisting of squares.

Fig. 5 shows the situation for a preview ratio of 15 and a GFOV of 5 degrees. Fig. 8 shows how a change in orientation of 1 degree affects the representation. The image has shifted over 20% of the total display area.

Fig. 6 shows the situation for a preview ratio of 1.6 and a GFOV of 70 degrees, and Fig. 7 for a preview ratio of 5.5 and a GFOV of 50 degrees. Fig. 9 shows how a 1 degree change in orientation affects the scene presented in Fig. 6 and Fig. 10 shows how a 1 degree change in orientation affects the scene presented in Fig. 7.

*For a square display, both HFOV and VFOV are equal to GFOV. Otherwise the HFOV is equal to the aspect ratio times VFOV.

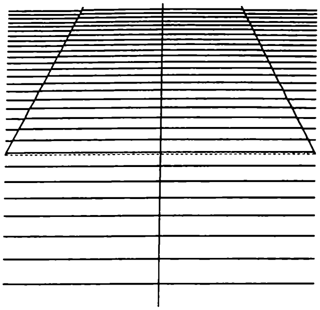


Fig. 5. GFOV=5 [deg], θ=-5 [deg].

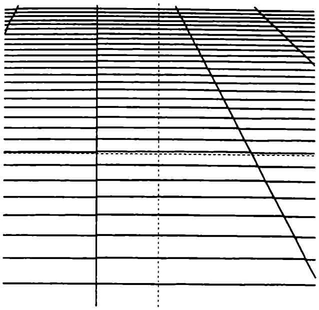


Fig. 8. GFOV=5 [deg], θ=-5 [deg].

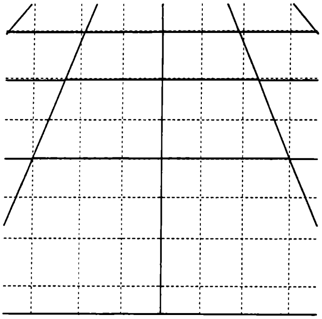


Fig. 6. GFOV=70 [deg], θ=-55 [deg].

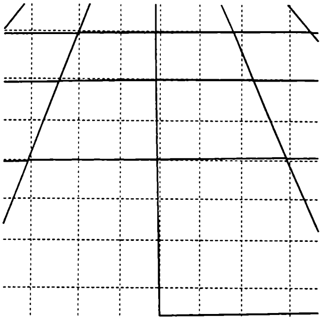


Fig. 9. GFOV=70 [deg], θ=-55 [deg].

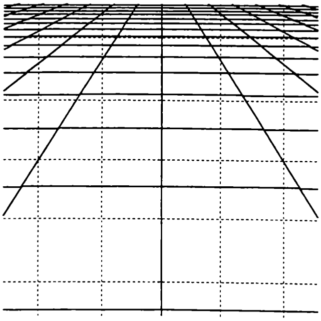


Fig. 7. GFOV=50 [deg], θ=-30 [deg].

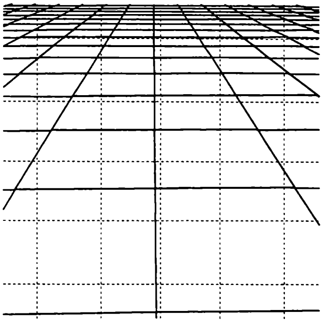


Fig. 10. GFOV=50 [deg], θ=-30 [deg].

Application

It will be illustrated how the method has been used to specify the design parameters for an exocentric display format to be used for surface movement guidance and control. This format was one of three potential format which were being considered for this application.
The starting point was a plan view with a range of 200m and a trajectory preview of 150m. Fig. 11 presents an example of this plan view.

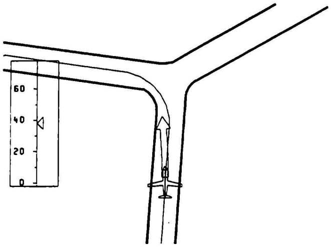


Fig. 11. Plan view display showing aircraft, taxiways, desired route, speed indicator and track predictor.

The design goal was to approximately double the range while at the same time increasing the position error resolution with at least 50%. The minimum ratio $R_{PREVIEW}$ becomes 3. Table 3 indicates the design space for the different elevations and compression ratios.

Table 3. Design space for different preview ratios

	2	3	4	5	6	7	8	9
-5	13	15	18	21	24	26	29	32
-10	6.5	7.6	9.0	10	12	13	14	16
-15	4.4	5.1	6.0	7.0	8.0	8.9	9.7	11
-20	3.3	3.9	4.6	5.3	6.0	6.7	7.3	8.2
-25	2.7	3.2	3.7	4.3	4.8	5.4	5.9	6.6
-30	2.3	2.7	3.1	3.6	4.1	4.6	5.0	5.5
-35	2.0	2.3	2.7	3.1	3.6	4.0	4.4	4.8
-40	1.8	2.1	2.4	2.8	3.2	3.5	3.9	4.3
-45	1.6	1.9	2.2	2.6	2.9	3.2	3.6	3.9
-50	1.5	1.8	2.0	2.4	2.6	X	X	X

The design space is also indicated in Table 4. Furthermore, to prevent a too fast scene rotation with a change in vehicle orientation, a lower limit of 30 degrees is selected for the GFOV. To limit the maximum perspective distortion to 20%, an upper limit of 70 degrees is imposed on the GFOV. The white area in Table 4 indicates the resulting design space.

Table 4. Design space for different GFOV's

	2	3	4	5	6	7	8	9
-5	3.3	5.0	6.0	6.8	7.3	7.5	7.8	8.0
-10	6.8	10	12	14	14	15	16	16
-15	10	15	18	20	22	23	24	24
-20	14	21	25	27	29	31	32	33
-25	18	26	31	35	37	39	40	41
-30	22	32	38	42	45	47	48	50
-35	26	39	46	50	53	56	57	59
-40	31	46	54	59	62	64	66	68
-45	37	53	62	68	71	74	76	77
-50	43	62	71	77	80	X	X	X

A potential design solution is a GFOV of 40 degrees and an elevation of -30 degrees. With the requirement that the visible range should be doubled to 400m, Eq. (1) yields a height of the viewpoint of 83m. As mentioned earlier, the visible area in front of the aircraft is typically of more importance than the area behind the aircraft. In Fig. 1 the angle ϕ_P was used to indicate the difference between the central viewing axis and the axis connecting the vehicle to the viewpoint. Fig. 12 shows the visible area and the position of ownship with a GFOV of 40 degrees, an elevation of -30 degrees, a viewpoint height of 83m and an angle ϕ_P which is set to 25% of the GFOV.

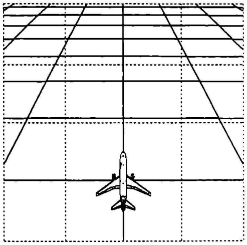


Fig. 12. GFOV=40 [deg], θ=-30 [deg], ϕ_P=10 [deg].

Setting the angle ϕ_P to 10 degrees results in a preview of 375m and a tail view of 30m. Fig. 13 shows the display format in which the aircraft is located on a taxiway.

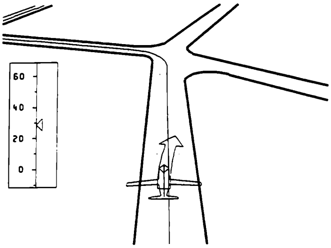


Fig. 13. Exocentric display format with GFOV=40 [deg], θ=-30 [deg], ϕ_P=10 [deg]. The presentation comprises the taxiways, the reference trajectory, a speed tape, and a track predictor.

Format design

The main question is *'How can the system best support the aquisition and control of information'*. With today's computer graphics technology it is no problem to generate an almost photorealistic depiction of the airport. The idea behind this approach is to support the taxi task with some kind of a synthetic vision system (SVS) which may include additional cues to support the navigation, guidance, and control task. Sachs et al.[6] describe SVS tests with a highly modified car which demonstrate that with the SVS approach high tracking performance can be achieved. The question is whether such a high level of realism is needed. As indicated by Owen[4], *'realism and fidelity have received more attention than they deserve, given how little is know about what specific information is needed to support the simulation of particular tasks'*. With respect to simulation training, Owen[4] concludes that *'where there are task-specific differences in the utility of an optical variable, pilots can also be trained to use the type of information that is optimal for each task. The choice depends on which performance variable is important as well'*. When designing a display for guidance and control, rather than training pilots to use the information that is optimal, a design goal is to provide only the information that is optimal.

Young et al.[8] describe a system which comprises an exocentric view for navigation purposes and an egocentric view for guidance and control. The design parameters of the egocentric view follow from the conformality requirements dictated by the head-up display. The exocentric view serves to provide the required navigation awareness, and as a result different requirements on position error resolution will result in different design parameters for the frame of reference. The design which will be discussed next employs an exocentric frame of reference for guidance and control purposes. Furthermore, it differs from the SVS approach in the following way. Rather than assuming that a high fidelity simulation of the real world will contain all cues which are needed to provide the required information, a representation will be specified which only contains the elements needed to convey task relevant information. Although one could argue that not enough is known about how all elements which can be perceived in the real world contribute to task performance, the following is true:

- The data needed for the control task can be obtained from a rather limited number of cues
- There is an asymptotic relation between the amount of cues and task performance.
- The likelihood of display clutter increases with the number of elements presented
- The fact that some cues may be additive but others only confirmatory[4] cautions against the presentation of too much 'reality'.

The representation of the taxiways provides information about the position constraints. A centerline is used to indicate the desired taxi route. Besides information about the position error, the pilot needs information about vehicle velocity. To provide accurate quantitative data, a speed tape is presented. In the initial evaluation, subjects mentioned that the display format did not provide any salient velocity cues. To increase the texture flow rate, equally spaced poles were included to the left and right of the centerline indicating the desired taxi route. Alphanumeric identifiers for runways and taxiways were also included. To support pilots with curve negotiation by allowing them to apply accurate open-loop control, a three second prediction of the future vehicle track is included. In contrast to single point position predictors used in perspective flightpath displays, this predictor showed the full track. The reasons for depicting the full track were:

1. Removal of position ambiguity
2. Providing information to check whether the track towards the future position violates position constraints.

With respect to the first reason, other possibilities exist which may produce less display clutter than the depiction of a complete track (e.q. the transparant window). However, the second reason for depiction of the track is that although the future position at a single time ahead may lie within the constraints, the track towards it can violate the constraints in an unacceptable way. For example, when a pilot has to turn onto a certain taxiway, a position predictor might show that the future position is on the desired taxiway, but as a result of corner cutting the runway constraints will be violated. In this situation it is important for the pilot to know that the track is not acceptable.

Evaluation

The motivation for an exocentric frame of reference is that it has the potential to combine a high position error gain with an increase in preview. The increase in position error gain should allow higher tracking accuracy while the improved preview increases the time horizon, thus allowing earlier detection of other traffic.
To obtain some initial feedback on the exocentric display format for taxi guidance, a prototype display format was implemented in a part-task simulator and an experiment was performed in which three different types of display formats, egocentric, exocentric, and planar were compared. The final part of the paper will discuss the experiment.

Experimental design

A 3x2 ANalysis Of VAriance (ANOVA) repeated measures within subject design was used for the experiment. The two factors were the frame of reference (3) and the predictor (2, disabled/enabled).
A map, which contained part of Amsterdam's Schiphol Airport, was used to choose six equally difficult routes, with a length varying between 4200 and 5900 meters. All tracks were made 22.5 meters wide. The routes were presented in a mixed order, each under different frame of reference and predictor conditions. Subjects completed each track in every possible condition. Using this extensive method of 36 test runs per subject, learning effects which remain after training should average out in the results.

Vehicle dynamics

The controls comprised a throttle lever box, brake pedals, and a sidestick. To simulate the aircraft's dynamics on the ground a standard vehicle model was adjusted to the weight and size of an average, mid-sized aeroplane (B737-400) to cover the inertia and handling aspects.
A software based nose wheel controller was developed. This controller simulates the latency effects caused by the side-stick to nose wheel servo transition. The angular velocity of the nose wheel varied with time according to an S-curved profile (Fig. 14).

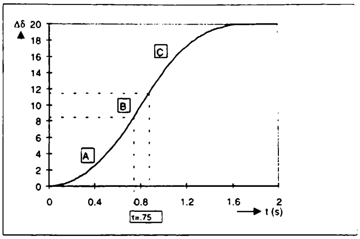


Fig. 14. Nosewheel deflection

The figure shows areas A, B and C. During the 'acceleration area' A, the angular velocity increases until it reaches its maximum value after seconds. In area B, changes at this constant, maximum rate. During the 'deceleration area' C, the angular velocity decreases to zero, leaving at the required angular change.

Display

To simulate a typical PFD, a CRT was covered to a size of 6.5" x 6.5". The simulation program displayed anti-aliased images at a refresh rate of 75 Hz. The update-rate of the display format was half the refresh rate (37.5 Hz). The plan view display format was the similar to the one presented in Fig. 11 and the exocentric format was similar to the one in Fig. 13. Besides the exocentric and the plan view format, an egocentric format was used. It was assumed that a egocentric format would allow the highest tracking performance to be achieved. Fig 15 presents the egocentric display format.

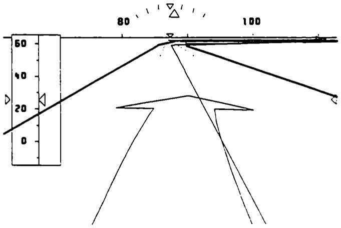


Fig. 15. Egocentric display format showing the taxiways, the reference trajectory, a speed tape, a heading tape, the deflection of the nosewheel, and a track predictor.

In contrast to the displacement cues of the exocentric and plan view displays, the dominant position error cue in the egocentric format is the change in splay angle[7]. The position error gain of the egocentric display is 2.11 [deg/m], and the orientation error gain is 2.5 [%/deg]. Each display format could be augmented with a track predictor.

Subjects and task

Six students from the Department of Electrical Engineering participated as subjects. Their instructions were to follow the desired trajectory as accurate as possible while maximizing the ground speed of the aircraft. Subjects were instructed to follow hold commands by stopping before red hold bars, and prevent collisions with other aircraft. Training session covered between 6 and 8 runs. Subjects drove with all display configurations to familiarize themselves with the different views and predictor conditions. The reference frame and track number were randomized during training.

Environment

During the experiment, several ground operation situations were simulated. During each run, 2 other aircraft drove around the airfield, having intersecting paths with the subject's route. It depended on the subject's speed whether, and if so, where the aircraft met each other. If a subject came within 200 meters of other aircraft, an event file was recorded. This file contained speed, position and control inputs of all aircraft involved in the event, which facilitated post experimental evaluation.

Measures

The data needed for the performance analysis was collected on each run at a rate of 37.5 Hz. It contained information on aircraft parameters and control inputs. During these runs, the maximum occurring cross track error was continuously monitored. If the error exceeded 10 meters, the run was repeated. Subjects could indicate if they were disturbed in any way during a run. In that case too, the run was repeated. The standard deviation of the cross track error (δXTE) served as a measure of tracking performance.

Results

hold commands: The stop bar projection delivered an adequate cue to all subjects. Every time it was projected, a safe distance to the bar was attended and the aircraft was successfully brought to a safe stop, in every frame of reference.

Conflict detection: Within every reference frame it seemed difficult at some points to avoid other aircraft crossing the subject's path. Especially vehicles approaching from either left or right of the aircraft often caused major problems. The ego-referenced view performed worst in this case, which could be expected from its limited GFOV. The best collision avoidance performance was achieved in the 3D exocentric view.

Tracking performance and velocity: Fig. 16 shows the mean and the standard deviation of δXTE and Fig. 17 shows the mean and the standard deviation of the velocity, both for straight segments. For curved segments, Fig. 18 shows the mean and the standard deviation of δXTE and Fig. 19 shows the mean and the standard deviation of the velocity. The results illustrate that on straight segments, tracking performance is better, and velocity is higher than on curved segments. Table 5 presents the results of the ANOVA.

Table 5. Summary of the results from the ANOVA. Results that are significant (α=0.05) are accentuated in grey.

	Ref. Frame	Prediction	Interaction

δXTE,s	F=39.36,
	p<0.0005
		F=2.07,
		p=0.152
			F=0.017,
			p=0.984

δXTE,c	F=3.36,
	p=0.036
		F=21.76,
		p=<0.0005
			F=33.34,
			p<0.0005

V,s	F=26.48,
	p<0.0005
		F=30.22,
		p<0.0005
			F=1.7,
			P=0.185

V,c	F=23.96, p<0.0005	F=120.46, p<0.0005	F=7.83, p<0.0005

Discussion

Conflict detection: given the fact that the exocentric frame of reference provided the largest visible area, the superiority in conflict detection was to be expected.

Tracking performance and velocity with the unaugmented display: On straight segments, the egocentric frame of reference yielded the best tracking performance. On curved segments this was not the case. With an egocentric frame of reference, while driving on a straight segment, subjects can extract position error information from the distortion of the symmetrical shape presented by the lines indicating the boarders of the taxiway. Orientation error information can be obtained from the displacement of the taxiway relative to the central vertical display axis. In a curve, there is no symmetrical reference, making the tracking task harder.

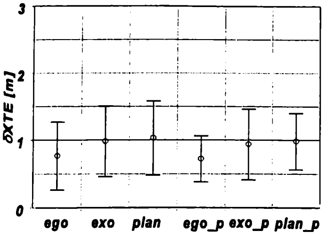


Fig. 16. Standard deviation of the cross track errors on straight segments.

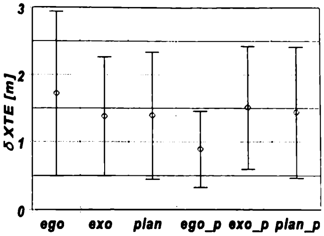


Fig. 18. Standard deviation of the cross track errors on curved segments.

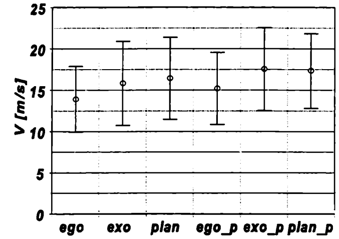


Fig. 17. Average velocity on straight segments.

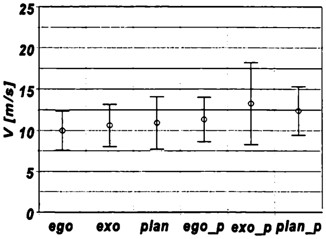


Fig. 19. Average velocity on curved segments.

With the exocentric frame of reference and the plan view display, the type of the position error cues is the same on straight and curved segments, i.e. the displacement of the aircraft symbol relative to the centerline. Only the difficulty of the tracking task is different between straight and curved segments. Orientation error cues are obtained from the relative orientation of the taxiway. In the experiment, only the magnitude of the position error cues was different between the latter two frames of reference. However, no significant difference in tracking performance was found between the exocentric display format and the plan view display. As indicated earlier, the design goal of the exocentric display format was to double the preview range relative to the plan view, and increase the position error gain with 50%. The results indicate that the increase in position error gain did not yield an improved tracking performance. Therefore, one might consider reducing the position error gain in order to achieve even more trajectory preview. With the approach presented in the first part of this paper, a method is available which allows the new design parameters to be specified in a structured way.

The results also show that the higher tracking performance with the egocentric display is achieved at a lower average velocity. This may be caused by the stronger velocity cues conveyed by the egocentric display. Furthermore, the higher gain and /or the different nature of the position and orientation error cues might influence the thresholds used by the subjects to determine when to reduce their velocity. A similar finding is reported by Lasswell and Wickens[2] who compared an egocentric perspective display with a plan view display (range 500 ft) and report a higher average taxi speed with the plan view display (14 kts 3-D vs 16 kts 2-D) on straight segments. They did not find any significant difference in lateral tracking performance, and did not analyze curved segments.

Tracking performance and velocity with the augmented display: When the position predictor was present, the tracking performance on curved segments improved significantly for the egocentric display format. This was to be expected since rather than having to extract position error-rate information, subjects could now use the deviation from the future reference trajectory as the primary source of information to close the position tracking loop. For the exocentric and plan view display, the predictor did not yield a significant difference in tracking performance, only an increase in average velocity. This increase in average velocity in curved segments is likely to be caused by the fact that the control task is easier. Thus, the predictor allows subjects to increase their average velocity without sacrificing tracking performance. Subjects also reported that they found the additional velocity cues presented through the length of the predicted track useful to control their velocity.

Conclusions

Perspective displays are not only useful for providing spatially integrated data for control in 3 spatial dimensions. With a 2-D control task, the additional degree of freedom can be used to define an exocentric frame of reference which combines a high position error gain with adequate trajectory preview much better than a plan view display. The values of the parameters which determine the frame of reference and the field of view should follow from task dictated visual cue requirements in 2-D display space. The proposed method can be used to prevent a trial and error process for the selection of an exocentric viewpoint. It does not generate a single best solution but uses task oriented requirements to determine the limits of the design space. This is considered an appropriate approach since it is regarded quite unrealistic to assume that all the design requirements are detailed enough to warrant the existence of a single best solution.

The results of the experiment with the three different frames of reference showed that for the tracking task, the drawbacks of the egocentric reference frame when tracking a curved segment can be compensated by including a prediction of the future vehicle track. They also showed that with respect to conflict detection the egocentric display performed the worst.

The fact that the increased position error resolution of the exocentric display relative to the plan view display did not yield an increase in tracking performance suggests that with the current exocentric display format a further increase in preview range is possible without having to sacrifice tracking performance. Finally, the results showed that the best collision avoidance performance was achieved with the 3D exocentric view.

Acknowledgments

The research into spatially integrated data presentation for navigation, guidance, and control is performed in the context of the Delft Program for Hybridized Instrumentation and Navigation Systems phase II (DELPHINS II), which is being sponsored by the Dutch Technology Foundation STW.

The experimental design, the implementation of the ground vehicle dynamics, and the data analysis for the evaluation of the different frames of reference was performed by Edwin Reinhoudt as part of his M.Sc. graduation work.

References

1. **Boyer, B.S. and Wickens, C.D.** (1994). *3D Weather Displays for Aircraft Cockpits.* ARL-94-11/NASA-94-4.
2. **Lasswell, J.W. and Wickens, C.D.** (1995). The Effects of Display Location and Dimensionality on Taxi-Way Navigation, ARL-95-5/NASA-95-2.
3. **Möhlenkamp, K. and Schänzer, G.** (1994). 'Automatic Control Steps for Aircraft Taxi Guidance', *Proceedings of the AGARD Meeting on Machine Intelligence in Air Traffic Management.*
4. **Owen, D.H.** (1996). 'Redundant Sources of Visual Information for Active Control: Additive or Confirmatory?'. *Proceedings of the 1996 IMAGE Conference*, June 23-28, Scottsdale, AZ.
5. **Prevett, T.T., Wickens, C.D.** (1994). *Perspective Displays and Frame of Reference: Their Interdependence to Realize Performance Advantages over Planar Displays in a Terminal Area Navigation Task*, Aviation Research Laboratory, University of Illinois at Urbana-Champaign, Il.
6. **Sachs, G., Möller, H., Dobler, K., Schänzer, G., and Möhlenkamp, K.** (1994). 'Synthetic Vision and Precision Navigation for Aircraft Taxi Guidance in Low Visibility', *Proceedings of the AIAA GNC Conference*, pp. 1202-1211, Scottsdale, AZ.
7. **Theunissen, E.** (1997). *Integrated Design of a Man-Machine Interface for 4-D Navigation*, ISBN 90-407-1406-1, Delft University Press, Delft, The Netherlands.
8. **Young, S.D., Jones, D.R., Bryant, W.H., and Eckhardt, D.E.** (1998). 'Safely improving airport surface capacity', *Aerospace America*, Vol. 36, No. 5, pp. 22-27.

AIAA-98-4175

A MODEL OF ATMOSPHERIC TURBULENCE FOR ROTORCRAFT SIMULATION AND ANALYSIS OF STABILITY AND PERFORMANCE

Mario G. Perhinschi[*]
Flight Dynamics and Control Laboratory
National Aerospace Research Institute
Bucharest, Romania

ABSTRACT

The purpose of this paper is to present the development of a simple turbulence representation suitable for helicopter dynamic simulation, stability and performance analysis, as well as pilot-in-the-loop rotorcraft handling qualities assessment. The frozen field hypothesis and linear approximation are assumed. Components of the uniform field velocity are included as additive perturbations affecting aerodynamic terms of the state matrix. Effects of the spatial gradients on parts of the helicopter other than rotor are considered as equivalent rotations. Effects of the spatial gradients on the rotor are considered by means of equivalent blade pitch angle perturbations. Vertical gradients are neglected. Equivalent blade pitch angle perturbations are determined such that total aerodynamic forces and moments over the rotor disc are conserved. The approach leads to a linear system with part of the input random. Some simulation results are presented to illustrate behavior and use of the model.

1. INTRODUCTION

Flight in turbulent atmosphere, as the most frequent poor weather condition, is a major issue for aircraft design. The most important problems related to turbulence effects are: structural loading and fatigue, controllability and handling qualities, passenger and crew safety and comfort. While structural analysis and design may require more complex and intricate descriptions of the natural phenomena, simulation, performance analysis or handling qualities investigations may take advantage of simpler and more tractable models.

The purpose of this paper is to present the development of a simple turbulence representation suitable for helicopter dynamic simulation, stability and performance analysis, as well as pilot-in-the-loop rotorcraft handling qualities assessment.

The frozen field hypothesis is usually assumed and the turbulence velocity is obtained by adding a spatial gradient to the random value at a fixed reference point. In the case of fixed-wing aircraft turbulence effects are included[1] as linear and angular perturbations affecting the aerodynamic terms in the equations of motion. Things are more complicated with rotary than with fixed-wing aircraft. The frozen field concept is still adopted and the spatially uniform component can still be included as an additive perturbation in the aerodynamic terms. Effects of spatial gradients on various parts of the helicopter can still be considered as equivalent rotations in a manner similar to the fixed wing but a different approach is needed to take into account for the effects of turbulence gradients on the rotor. Hess[2] converted lateral gradients of the turbulence vertical velocity into longitudinal cyclic inputs. This approach is generalized and gradients in both horizontal directions of all three components of the turbulence velocity vector are considered and modeled as equivalent blade pitch angle perturbations matched to preserve the same aerodynamic forces and moments over the entire rotor disc. A linear model of the helicopter in turbulence is built with turbulence velocity components, equivalent rotations and equivalent blade pitch angle perturbations as random inputs.

2. LOCAL AIR VELOCITY EXPRESSION

Considering the linear field approximation, let the total air velocity vector due to turbulence $\bar{V}_g$ at any point in space be the sum of the turbulence air velocity vector at a fixed reference point (a spatially uniform component) and a spatial gradient:

$$\bar{V}_g = \bar{V}_{gu} + \bar{V}_{gg} \qquad (1)$$

[*] Senior researcher

Then the components in body axes of the total air velocity vector $\bar{V}_g$ are given by equation (2). Considering the geometry of the rotorcraft and the characteristics of aerodynamic phenomena the vertical gradients will be neglected. The flapping angle will be considered small and the effects of air flow along the blade neglected. The lateral attitude angle of the rotorcraft is assumed zero. Therefore the resultant components of air velocity "seen" by a blade section at an arbitrary location r_b, along the blade are shown in Figure 1. Let the x axis of the blade related frame be along the blade projection on the hub plane, positive towards the hub center while the y axis lies in the hub plane, positive against rotor angular velocity. Then the expressions of the total air velocity components in the blade related frame are given by equation (3).

$$[V_g]_B = [V_{gu}]_B + [V_{gg}]_B = \begin{bmatrix} u_{gu} \\ v_{gu} \\ w_{gu} \end{bmatrix} + \begin{bmatrix} u_{gg} \\ v_{gg} \\ w_{gg} \end{bmatrix} =$$

$$= \begin{bmatrix} u_{gu} \\ v_{gu} \\ w_{gu} \end{bmatrix} + \begin{bmatrix} \frac{\partial u_g}{\partial x} & \frac{\partial u_g}{\partial y} & \frac{\partial u_g}{\partial z} \\ \frac{\partial v_g}{\partial x} & \frac{\partial v_g}{\partial y} & \frac{\partial v_g}{\partial z} \\ \frac{\partial w_g}{\partial x} & \frac{\partial w_g}{\partial y} & \frac{\partial w_g}{\partial z} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (2)$$

$$[V_a]_b = -V \begin{bmatrix} 0 \\ -\cos\theta \sin\psi_b \\ \sin\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w_i \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega r_b \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \dot{\beta} r_b \end{bmatrix} +$$

$$+ \begin{bmatrix} 0 \\ -u_{gu}\sin\psi_b + v_{gu}\cos\psi_b \\ w_{gu} \end{bmatrix} + \begin{bmatrix} 0 \\ v_{gg} \\ w_{gg} \end{bmatrix} \qquad (3)$$

Let us denote:

$$v_{gg} = r_b[(\frac{\partial u_g}{\partial x} - \frac{\partial v_g}{\partial y})\sin\psi_b \cos\psi_b + \frac{\partial u_g}{\partial y}\sin^2\psi_b - \frac{\partial v_g}{\partial x}\cos^2\psi_b] \qquad (4)$$

$$w_{gg} = -r_b(\frac{\partial w_g}{\partial x}\cos\psi_b + \frac{\partial w_g}{\partial y}\sin\psi_b) \qquad (5)$$

$$\bar{V} = V\cos\theta - u_{gu} \qquad (6)$$

$$\bar{w} = w_i + w_{gu} - V\sin\theta \qquad (7)$$

Then the modulus of the air velocity is given by the following expression:

$$|\bar{V}_a| = [(\bar{V}\sin\psi_b + v_{gu}\cos\psi_b + \Omega r_b + v_{gg})^2 + (\bar{w} + \dot{\beta} r_b + w_{gg})^2]^{1/2} \qquad (8)$$

3. EQUIVALENT BLADE PITCH ANGLE

On a blade element of width dr_b, at current location r_b along the blade and azimuth angle ψ_b, the elementary lift in still air is:

$$dL(\psi_b, r_b) = \frac{\rho}{2} c U^2 C_{L\alpha} \alpha dr_b \qquad (9)$$

It will be assumed that the turbulence gradients will generate a variation of local air speed ΔU and a variation of local blade angle of attack $\Delta\alpha$ such that the elementary lift is:

$$dL_t(\psi_b, r_b) = \frac{\rho}{2} c (U + \Delta U)^2 C_{L\alpha} (\alpha + \Delta\alpha) dr_b \qquad (10)$$

It has been observed[2] that the lateral gradient of the vertical component of the velocity will determine effects similar to those of longitudinal cyclic angle perturbations, while the longitudinal gradient of the same component of the velocity will determine effects similar to those of lateral cyclic perturbations, due to rotor dynamic response. Based on these observations we want to equivalate the effect of horizontal gradients of all three components of the velocity vector with a variation of local blade angle of attack $\Delta\alpha_e$ such that total aerodynamic forces and moments over the rotor disc are conserved, that is:

$$\int_0^R \int_0^{2\pi} dL_t = \int_0^R \int_0^{2\pi} dL_e \qquad (11)$$

The elementary equivalent lift is expressed as:

$$dL_e(\psi_b, r_b) = \frac{\rho}{2} c U^2 C_{L\alpha} (\alpha + \Delta\alpha_e) dr_b \qquad (12)$$

The variation of the local blade angle of attack $\Delta\alpha_e$ is expressed in terms of equivalent perturbations of collective, lateral and longitudinal cyclic angles:

$$\Delta\alpha_e = \delta\theta_0 - \delta A_{1H}\cos\psi_b - \delta B_{1H}\sin\psi_b \qquad (13)$$

Local air velocity and blade angle of attack without turbulence gradients effect are given by the following expressions (see also Figure 1):

$$U^2 = (\bar{V}\sin\psi_b + v_{gu}\cos\psi_b + \Omega r_b)^2 + (\bar{w} + \dot{\beta} r_b)^2 \qquad (14)$$

$$\alpha = \theta_0 - A_{1H}\cos\psi_b - B_{1H}\sin\psi_b - \\ - a\tan\frac{\overline{w} + \dot{\beta} r_b}{\overline{V}\sin\psi_b + v_{gu}\cos\psi_b + \Omega r_b} \tag{15}$$

Local air velocity and blade angle of attack with turbulence gradients effect are given by the following expressions:

$$(U + \Delta U)^2 = (\overline{V}\sin\psi_b + v_{gu}\cos\psi_b + \Omega r_b + v_{gg})^2 + \\ + (\overline{w} + \dot{\beta} r_b + w_{gg})^2 \tag{16}$$

$$\alpha + \Delta\alpha = \theta_0 - A_{1H}\cos\psi_b - B_{1H}\sin\psi_b - \\ - a\tan\frac{\overline{w} + \dot{\beta} r_b + w_{gg}}{\overline{V}\sin\psi_b + v_{gu}\cos\psi_b + \Omega r_b + v_{gg}} \tag{17}$$

In the context of the frozen field approach, the spatial turbulence gradients can be converted to time gradients by considering that:

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial t}\frac{\partial t}{\partial x} = \frac{1}{V_{rx}}\frac{\partial}{\partial t} \tag{18}$$

$$\frac{\partial}{\partial y} = \frac{\partial}{\partial t}\frac{\partial t}{\partial y} = \frac{1}{V_{ry}}\frac{\partial}{\partial t} \tag{19}$$

The reference velocities V_{rx} and V_{ry} represent rates at which turbulence perturbations succeed. If the helicopter is moving then they may be taken equal to the flight velocity. To extend the approach to hover some values have to be chosen according to the level of turbulence^2.

Let us substitute equations (13), (14) and (15) in (12), and (16) and (17) in (10). After some manipulations and approximations, integrals are solved and the equivalence equation (11) becomes equation (20):

$$\begin{aligned}
&\{\frac{1}{R}[\overline{V}^2 + v_{gu}^2 + 2\overline{w}^2] + \frac{2R}{3}\Omega_r^2[1 + \frac{1}{2}(\beta_c^2 + \beta_s^2)]\}\delta\theta_0 - \\
&\quad - \Omega_r[(\overline{V} + \overline{w}\beta_c)\delta B_{1H} - (v_{gu} - \overline{w}\beta_s)\delta A_{1H}] = \\
&= \overline{w}\dot{w}_g(V_{rx}^{-1}A_{1H} + V_{ry}^{-1}B_{1H}) - \\
&\quad - \frac{\overline{V}}{4}[(\dot{u}_g V_{rx}^{-1} - \dot{v}_g V_{ry}^{-1})A_{1H} + (3\dot{u}_g V_{ry}^{-1} - \dot{v}_g V_{rx}^{-1})B_{1H}] - \\
&\quad - \frac{v_{gu}}{4}[(\dot{u}_g V_{ry}^{-1} - 3\dot{v}_g V_{rx}^{-1})A_{1H} + (\dot{u}_g V_{rx}^{-1} - \dot{v}_g V_{ry}^{-1})B_{1H}] + \\
&\quad + \frac{R}{3}\theta_0[2\Omega_r\dot{w}_g(\beta_c V_{ry}^{-1} - \beta_s V_{rx}^{-1}) + \\
&\quad + 2\Omega_r(\dot{u}_g V_{ry}^{-1} - \dot{v}_g V_{rx}^{-1}) + \dot{w}_g^2(V_{rx}^{-2} + V_{ry}^{-2}) + \\
&\quad + \frac{1}{4}(\dot{u}_g V_{rx}^{-1} - \dot{v}_g V_{ry}^{-1})^2 + \frac{3}{4}\dot{u}_g^2 V_{ry}^{-2} + \frac{3}{4}\dot{v}_g^2 V_{rx}^{-2} - \\
&\quad - \frac{1}{2}\dot{u}_g\dot{v}_g V_{rx}^{-1}V_{ry}^{-1}] - \\
&\quad - \frac{1}{2}[\overline{w}(\dot{u}_g V_{ry}^{-1} - \dot{v}_g V_{rx}^{-1}) - \dot{w}_g(\overline{V}V_{ry}^{-1} + v_{gu}V_{rx}^{-1})] - \\
&\quad - \frac{\dot{w}_g}{\Omega}\{3\overline{w}\,(2\Omega_r\beta_s + \dot{w}_g V_{rx}^{-1})V_{rx}^{-1} + \\
&\quad + \frac{1}{2(\overline{V}^2 + v_{gu}^2)}\{-6\overline{V}\overline{w}v_{gu}\Omega(\beta_c V_{rx}^{-1} - \beta_s V_{ry}^{-1}) - \\
&\quad - \frac{3}{2}(v_{gu}^2 + 3\overline{V}^2)\overline{w}[\dot{w}_g(V_{rx}^{-2} - V_{ry}^{-2}) + \\
&\quad + 2\Omega(\beta_s V_{rx}^{-1} + \beta_c V_{ry}^{-1})]\}\}
\end{aligned} \tag{20}$$

The contributions of the turbulence gradients to air velocity expressed by equations (4) and (5) are plotted in Figure 2. Based on these, the effects of the lateral gradient of the vertical velocity are considered to be equivalent to a longitudinal cyclic perturbation, the effects of the longitudinal gradient of the vertical velocity are considered to be equivalent to a lateral cyclic perturbation and the effects of the gradients of the longitudinal and lateral velocities are considered to be equivalent to collective pitch perturbations. Neglecting some coupling terms we obtain for the general case:

$$\begin{aligned}
\delta B_{1H} &= [-\Omega_r(\overline{V} + \overline{w}\beta_c)]^{-1}\dot{w}_g V_{ry}^{-1}\{[\overline{w}B_{1H} + \\
&\quad + \frac{R}{3}\theta_0(2\Omega_r\beta_c + \dot{w}_g V_{ry}^{-1}) + \frac{1}{2}\overline{V}] - \\
&\quad - \frac{1}{2\Omega(\overline{V}^2 + v_{gu}^2)}\{6\overline{V}\overline{w}v_{gu}\Omega\beta_s + \\
&\quad + \frac{3}{2}(v_{gu}^2 + 3\overline{V}^2)\overline{w}(\dot{w}_g\ V_{ry}^{-1} - 2\Omega\beta_c)]
\end{aligned} \tag{21}$$

$$\begin{aligned}
\delta A_{1H} &= [-\Omega_r(v_{gu} - \overline{w}\beta_s)]^{-1}\dot{w}_g V_{rx}^{-1}\{[\overline{w}A_{1H} - \\
&\quad - \frac{R}{3}\theta_0(2\Omega_r\beta_s - \dot{w}_g V_{rx}^{-1}) + \frac{1}{2}v_{gu}] - \\
&\quad - \frac{1}{\Omega}\{3\overline{w}\,(2\Omega_r\beta_s + \dot{w}_g V_{rx}^{-1}) + \\
&\quad + \frac{1}{2(\overline{V}^2 + v_{gu}^2)}\{-6\overline{V}\overline{w}v_{gu}\Omega(\beta_c) - \\
&\quad - \frac{3}{2}(v_{gu}^2 + 3\overline{V}^2)\overline{w}[\dot{w}_g(V_{rx}^{-1}) + 2\Omega(\beta_s V_{rx}^{-1})]\}\}
\end{aligned} \tag{22}$$

$$\begin{aligned}\delta\theta_0 = \{&\frac{1}{R}(\overline{V}^2 + v_{gu}^2 + 2\overline{w}^2) + \frac{2R}{3}\Omega_r^2[1 + \frac{1}{2}(\beta_c^2 + \beta_s^2)]\}^{-1} \cdot \\ &\cdot \{-\frac{\overline{V}}{4}[(\dot{u}_g V_{rx}^{-1} - \dot{v}_g V_{ry}^{-1})A_{1H} + (3\dot{u}_g V_{ry}^{-1} - \\ &- \dot{v}_g V_{rx}^{-1})B_{1H}] - \frac{v_{gu}}{4}[(\dot{u}_g V_{ry}^{-1} - 3\dot{v}_g V_{rx}^{-1})A_{1H} + \\ &+ (\dot{u}_g V_{rx}^{-1} - \dot{v}_g V_{ry}^{-1})B_{1H}] + \frac{R}{3}\theta_0[2\Omega_r(\dot{u}_g V_{ry}^{-1} - \\ &- \dot{v}_g V_{rx}^{-1}) + \frac{1}{4}(\dot{u}_g V_{rx}^{-1} - \dot{v}_g V_{ry}^{-1})^2 + \frac{3}{4}\dot{u}_g^2 V_{ry}^{-2} + \\ &+ \frac{3}{4}\dot{v}_g^2 V_{rx}^{-2} - \frac{1}{2}\dot{u}_g \dot{v}_g V_{rx}^{-1} V_{ry}^{-1}] - \frac{1}{2}\overline{w}(\dot{u}_g V_{ry}^{-1} - \dot{v}_g V_{rx}^{-1})\}\end{aligned} \tag{23}$$

Further simplification of these expressions is possible because some of the terms are negligible. This formulation will allow to build up a linear system describing the dynamics of the rotorcraft including the effects of turbulence. Some of the inputs to this system are random.

If only turbulence on the vertical axis is considered then the expressions of the equivalent blade pitch angles become, for hover flight:

$$\begin{aligned}\delta B_{1H} = &- V_{ry}^{-1}[\frac{B_{1H}}{\Omega_r\beta_c} + \frac{2\theta_0 R}{3(w_i + w_{gu})} + \frac{3}{\Omega_r}]\dot{w}_g - \\ &- \frac{V_{ry}^{-2}}{\Omega_r\beta_c}[\frac{\theta_0 R}{3(w_i + w_{gu})} - \frac{3}{2\Omega_r}]\dot{w}_g^2\end{aligned} \tag{24}$$

$$\begin{aligned}\delta A_{1H} = &- V_{rx}^{-1}[\frac{A_{1H}}{\Omega_r\beta_s} - \frac{2\theta_0 R}{3(w_i + w_{gu})} + \frac{3}{\Omega_r}]\dot{w}_g + \\ &+ \frac{V_{rx}^{-2}}{\Omega_r\beta_s}[\frac{\theta_0 R}{3(w_i + w_{gu})} - \frac{3}{2\Omega_r}]\dot{w}_g^2\end{aligned} \tag{25}$$

$$\delta\theta_0 = 0 \tag{26}$$

If only turbulence on the longitudinal axis is considered then the expressions of the equivalent blade pitch angles become for hover flight:

$$\delta B_{1H} = 0 \tag{27}$$

$$\delta A_{1H} = 0 \tag{28}$$

$$\begin{aligned}\delta\theta_0 = \{&\frac{1}{R}(u_{gu}^2 + 2w_i^2) + \frac{2R}{3}\Omega_r^2[1 + \frac{1}{2}(\beta_c^2 + \beta_s^2)]\}^{-1}\dot{u}_g \cdot \\ &\cdot [\frac{u_{gu}}{4}(V_{rx}^{-1}A_{1H} + 3V_{ry}^{-1}B_{1H}) + \frac{R}{3}\theta_0(2\Omega_r V_{ry}^{-1} + \\ &+ \frac{1}{4}\dot{u}_g V_{rx}^{-2} + \frac{3}{4}\dot{u}_g V_{ry}^{-2}) - \frac{1}{2}w_i V_{ry}^{-1}]\end{aligned} \tag{29}$$

Finally, if only turbulence on the lateral axis is considered, then the equivalent cyclic angles are zero again but the equivalent collective pitch angle is given by the expression:

$$\begin{aligned}\delta\theta_0 = \{&\frac{1}{R}(v_{gu}^2 + 2w_i^2) + \frac{2R}{3}\Omega_r^2[1 + \frac{1}{2}(\beta_c^2 + \beta_s^2)]\}^{-1}\dot{v}_g \cdot \\ &\cdot [\frac{v_{gu}}{4}(3V_{rx}^{-1}A_{1H} + V_{ry}^{-1}B_{1H}) - \frac{R}{3}\theta_0(2\Omega_r V_{rx}^{-1} + \\ &+ \frac{1}{4}\dot{v}_g V_{ry}^{-2} - \frac{3}{4}\dot{v}_g V_{rx}^{-2}) + \frac{1}{2}w_i V_{rx}^{-1}]\end{aligned} \tag{30}$$

4. HELICOPTER MODEL IN TURBULENCE

We have assumed the linear approximation of the turbulence field, therefore the air velocity due to turbulence at any point is the superposition of a random value at a fixed reference point and a spatial gradient. Components of the uniform field u_{gu}, v_{gu}, w_{gu} are included as additive perturbations affecting aerodynamic terms of the state matrix. Effects of spatial gradients are included as follows.

4.1. Gradients Effects on Main Rotor

The lateral gradient of the vertical component of the air velocity is equivalent to a longitudinal cyclic angle perturbation δB_{1H}. The longitudinal gradient of the vertical component of the air velocity is equivalent to a lateral cyclic angle perturbation δA_{1H}. The lateral and longitudinal gradients of the horizontal components of the air velocity are equivalent to a collective blade pitch angle perturbation $\delta\theta_0$. Expressions for these equivalent perturbations are given by equations (21), (22) and (23), respectively for the general case. Expressions for single axis turbulence effects are also determined (equations (24), (25), (26) for vertical axis, (27), (28), (29) for longitudinal axis, (27), (28), (30) for lateral axis).

4.2. Gradients Effects on Parts of the Helicopter

Effects of the vertical velocity gradients on other main parts of the helicopter (fuselage, tail rotor, horizontal and vertical stabilizers, wing) are included as equivalent rotations in a manner similar to fixed wing aircraft[1]:

$$p_g = \frac{\partial w_g}{\partial y} \quad \text{and} \quad q_g = -\frac{\partial w_g}{\partial x} \tag{31}$$

4.3. Gradients Effects on Tail Rotor and Vertical Stabilizer

The longitudinal gradient of the lateral velocity component has an effect similar to that of a yawing rotation[1]:

$$r_{gV} = \frac{\partial v_g}{\partial x} \tag{32}$$

Let us assume that the longitudinal distances to the center of gravity from the tail rotor axis and from the vertical stabilizer center of pressure coincide and are denoted by l_T. At this location the longitudinal gradient of the longitudinal air velocity component will give a perturbation of the local velocity that can be expressed as:

$$u_{gV} = \frac{\partial u_g}{\partial x} l_T \tag{33}$$

4.4. Gradients Effects on the Wing and Horizontal Stabilizer

The lateral gradient of the longitudinal velocity component has an effect similar to that of a yawing rotation on the wing. If there is a horizontal stabilizer of significant span it can be treated as a wing. For unswept wing[1]:

$$r_{gA} = \frac{\partial u_g}{\partial y} \tag{34}$$

4.5. Linear System of the Helicopter in Turbulence

Assume that the helicopter linear system in still atmosphere is:

$$\dot{X} = AX + Bu_c \tag{35}$$

With usual notations, the state vector and the control vector as perturbations, are, respectively:

$$X = [\Delta u \;\; \Delta w \;\; \Delta q \;\; \Delta\theta \;\; \Delta v \;\; \Delta p \;\; \Delta\varphi \;\; \Delta r]^T \tag{36}$$

$$u_c = [\Delta\delta_c \;\; \Delta B_1 \;\; \Delta\delta_T \;\; \Delta A_1]^T \tag{37}$$

Operating all manipulations to include turbulence effects as described in previous paragraphs, the linear system can be written as:

$$\dot{X} = AX + B_t u_t \tag{38}$$

where the input vector becomes:

$$u_t = \begin{bmatrix} \begin{bmatrix} \Delta\delta_c \\ \Delta B_1 \\ \Delta\delta_T \\ \Delta A_1 \end{bmatrix}^T & \begin{bmatrix} u_{gu} \\ v_{gu} \\ w_{gu} \\ u_{gV} \end{bmatrix}^T & \begin{bmatrix} p_g \\ q_g \\ r_{gA} \\ r_{gV} \end{bmatrix}^T & \begin{bmatrix} \delta\theta_0 \\ \delta B_{1H} \\ \delta A_{1H} \end{bmatrix}^T \end{bmatrix}^T \tag{39}$$

Let us denote by V(m:n) a vector that consists of the elements m through n of vector V. A(:,i) and A(i,:) represent the column and respectively the row number i of matrix A while A(k:l,m:n) is a matrix whose elements are the elements a_{ij} of matrix A such that $k \le i \le l$ and $m \le j \le n$. With these notations the control matrix is:

$$B_t = [B \,\vdots\, A_{tV} \,\vdots\, A_{t\omega} \,\vdots\, B(:,1) \,\vdots\, B(:,2) \,\vdots\, B(:,4)] \tag{40}$$

where:

$$A_{tV} = -[\overline{A}(:,1) \;\; \overline{A}(:,5) \;\; \overline{A}(:,2) \;\; \overline{A}_{VT}(:,1)] \tag{41}$$

$$A_{t\omega} = -[\overline{A}_O(:,6) \;\; \overline{A}_O(:,3) \;\; \overline{A}_A(:,8) \;\; \overline{A}_{VT}(:,8)] \tag{42}$$

Matrix $\overline{A}$ is obtained from matrix A by excluding all non-aerodynamic terms. Matrix $\overline{A}$ affected by a subscript means that only contributions of specific parts of the helicopter are considered, as implied by the particular subscript. Index O stands for all parts other than main rotor. Index A represents contribution of the wing and horizontal stabilizer. Index VT represents the summed effects of vertical stabilizer and tail rotor. For example:

$$\overline{A}_{VT}(:,8) = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T & \begin{bmatrix} \dfrac{(Y_r)_{VT}}{m} \\ \dfrac{I_z(L_r)_{VT} + I_{xz}(N_r)_{VT}}{I_c} \\ 0 \\ \dfrac{I_{xz}(L_r)_{VT} + I_x(N_r)_{VT}}{I_c} \end{bmatrix}^T \end{bmatrix}^T \tag{43}$$

$$\begin{aligned} (Y_r)_{VT} &= (Y_r)_V + (Y_r)_T \\ (L_r)_{VT} &= (L_r)_V + (L_r)_T \\ (N_r)_{VT} &= (N_r)_V + (N_r)_T \end{aligned} \tag{37}$$

We have thus obtained a linear model with random inputs u_{gu}, v_{gu}, w_{gu} that can be generated in time domain[3] to match statistical properties of turbulence models[4,5] (i.e. von Karman, Dryden). Then equivalent rotations (p_g, q_g, r_{gA}, r_{gV}) and local longitudinal velocity perturbation at tail (u_{gV}), as well as equivalent collective pitch and equivalent cyclic angles can be computed. In this form the model can be used for simulation and stability and performance analysis. The addition of a human pilot model makes it suitable for handling qualities assessment[2,6].

5. NUMERICAL SIMULATION RESULTS

A structural human pilot model has been implemented to control system (38). A linear model of the helicopter IAR-330 Puma, a 7000 kg aircraft with single, four bladed main rotor and conventional configuration was used. Simple, single axis tasks at hover were considered. Commands are zero attitude angles and zero vertical velocity to reject turbulence perturbations of severe intensity at low altitude[5]. Von Karman turbulence model has been implemented.

To illustrate behavior and use of the model we present in Figures 3 to 12 some simulation results. Figure 3 shows variation of the generated turbulence velocity on the longitudinal axis together with the helicopter longitudinal velocity component in uniform field. Behavior in uniform field plus gradients is shown in Figure 4. Similar results for the lateral axis are presented in Figures 5 and 6 and for the vertical axis, in Figures 7 and 8. Variations of attitude angles and vertical helicopter velocity, with and without spatial gradients effects are presented in Figures 9 to 12.

6. CONCLUSIONS

A simple turbulence representation suitable for rotorcraft simulation, stability and performance analysis and pilot-in-the-loop handling qualities assessment has been developed resulting in a linear model with random inputs.

The model is based on the linear, frozen turbulence field concept extended to rotorcraft in forward flight and in hover.

Spatial gradients in both horizontal directions of all three components of the turbulence velocity vector are considered and modelled as equivalent collective, longitudinal and lateral cyclic perturbations to preserve the same aerodynamic forces and moments over the entire rotor disc.

Some illustrative simulation results for hover flight are presented but further conclusions, refinement and validation of the approach still need to be substantiated by comparison with experimental data which have not been available at this moment.

7. REFERENCES

1. **Etkin B.**: "Turbulent Wind and Its Effect on Flight," Journal of Aircraft, No. 5 1981

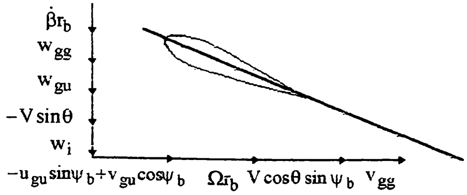
2. **Hess R. A.**: "Rotorcraft Handling Qualities in Turbulence," Proc. of the AIAA Atmospheric Flight Mechanics Conference, AIAA 93-3666, 1993, Monterey, USA

3. **Beal T. R.**: "Digital Simulation of Atmospheric Turbulence for Dryden and von Karman Models," Journal of Guidance, Control and Dynamics, No. 1, 1993

4. **Etkin B.**: "Dynamics of Flight - Stability and Control," John Wiley & Sons, New York, London, 1959

5. **Anon.**: "Military Specification: Flying Qualities of Piloted Airplanes," MIL-F-8785C, 1980

6. **Perhinschi M. G.**: "A Study of Helicopter Handling Qualities in Turbulence Using a Human Pilot Structural Model," AIAA Atmospheric Flight Mechanics Conference, Boston, 1998, AIAA paper 98-4148



Not actual scale

Figure 1. Air velocity components at local blade element

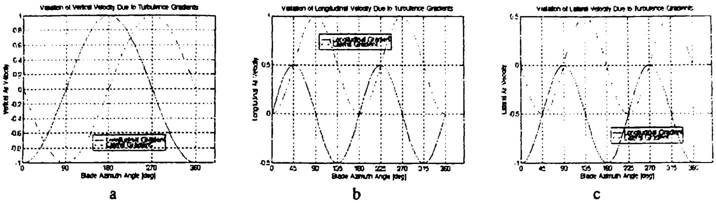


Figure 2. Variations of Velocity Components Due to Turbulence Gradients for a Complete Blade Rotation - (a) Vertical, (b) Longitudinal, (c) Lateral

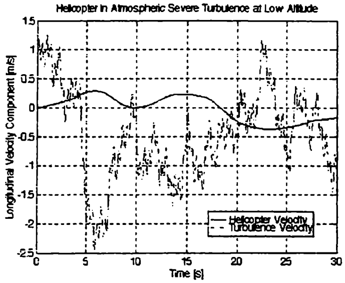


Figure 3. Helicopter Longitudinal Velocity in Uniform Turbulence Field

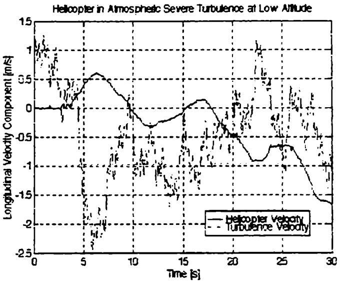


Figure 4. Helicopter Longitudinal Velocity in Uniform Turbulence Field with Gradients Effects

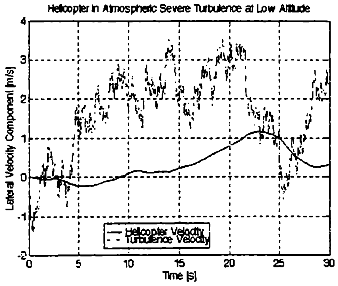


Figure 5. Helicopter Lateral Velocity in Uniform Turbulence Field

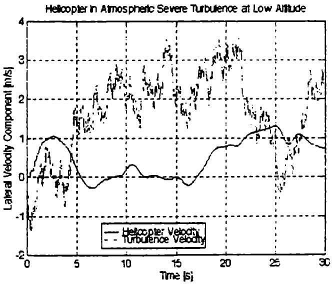


Figure 6. Helicopter Lateral Velocity in Uniform Turbulence Field with Gradients Effects

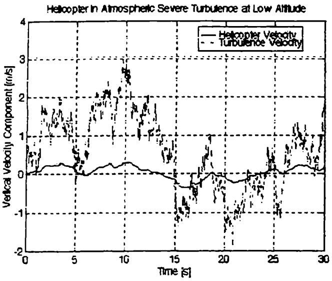


Figure 7. Helicopter Vertical Velocity in Uniform Turbulence Field

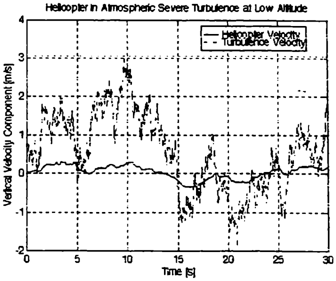


Figure 8. Helicopter Vertical Velocity in Uniform Turbulence Field with Gradients Effects

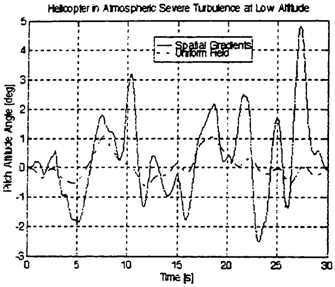


Figure 9. Pitch Attitude Angle in Uniform Turbulence Field with and without Gradients Effects

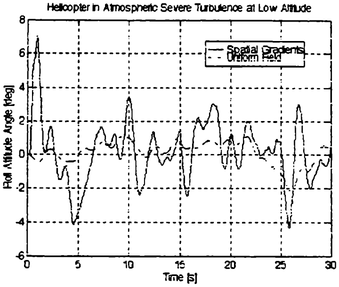


Figure 10. Roll Attitude Angle in Uniform Turbulence Field with and without Gradients Effects

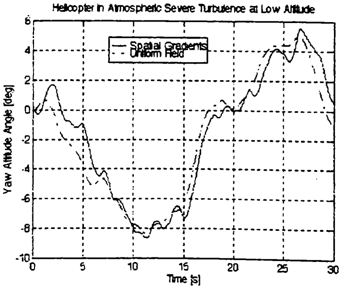


Figure 11. Yaw Attitude Angle in Uniform Turbulence Field with and without Gradients Effects

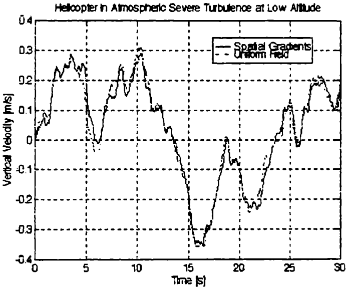


Figure 12. Vertical Velocity Component in Uniform Turbulence Field with and without Gradients Effects

AIAA-98-4176

TOWARD MINIMAL LATENCY SIMULATION SYSTEMS*

Ronald G. Moore
Senior Systems Engineer
Evans & Sutherland
Salt Lake City, UT
rmoore@es.com

Charles N. Pope
Electronics Engineer
Naval Air Warfare Center
Training Systems Division
Orlando, FL
charles_pope@ntsc.navy.mil

Eric Foxlin
Chief Technical Officer
InterSense, Inc.
Burlington, MA
ericf@isense.com

ABSTRACT

The careful integration of new hybrid tracking sensors and improved motion prediction algorithms, along with advanced image generation features such as pre-draw insertion of sensor data and image shifting, can provide simulation system developers with far greater flexibility (using COTS hardware) to find the optimal balance between cost, database complexity, display resolution, and acceptable levels and types of image artifacts without compromising system latency or accuracy. Personal Computer based data acquisition boards used in conjunction with the Electronic Visual Display Attitude Sensor (EVDAS) has led to the development of a comprehensive and portable latency measurement evaluation system for simulator troubleshooting, and for verifying that latency minimization goals are achieved. Minimal latency systems are not only needed in order to more fully meet new military training goals of greater fidelity, affordability, and deployability, but to transfer new technology to commercial and private use markets without health and safety risks and their concomitant liability.

INTRODUCTION

Latency was rated as the most prevalent source of simulator sickness by a panel of leading experts at the 7th International Conference on Human Computer Interaction.[1] Latency is but one of many causes of simulator side effects, however, as will be discussed this paper, it is increasingly a source over which simulator designers have considerable ability to control.

The issue of latency in high fidelity flight simulators was considered put to rest by 1990.[15] Pilots expect and actively seek a specific amount of delay in flight control response. In flight simulators with a 60 Hz update rate (corresponding to 5-6 frames or 83.3-100.0 milliseconds of end-to-end delay), effective latency does not need to be reduced dramatically in order to closely match the handling qualities response of a particular aircraft and, therefore, can be accomplished through straightforward predictive filtering.

Nevertheless, dome display facilities with multi-channel 60 Hz image generation equipment are out of the reach of most training system budgets. Moreover, requirements for part-task and deployable trainers will continue to make lower update rates and the use of head tracked display architectures attractive alternatives. Until recently, low update rates and head tracked imagery were not compatible solutions. The effects of latency are far more severe in head tracked systems, and are only compounded by lower update rates. In an HMD (as well as head mounted projector), all latency translates into unwanted image "swim." In head-mounted Augmented Reality displays, latency also manifests itself in a misregistration of the synthetic image overlaying the real world object of interest.[32-34] Area of Interest (AOI) systems in which the projector is not mounted on the head are not susceptible to these same latency effects. However, depending on the field of view of the inset and the head motion rates inherent to the task, latency may still need to be tightly controlled.

Increases in HMD fidelity have been slow due to the present resolution limits of miniature CRT's and Flat Panel Displays (FPD's). HMD's and compact projection systems will ultimately benefit from high resolution miniature displays based on Micro-Electro-Mechanical Systems ("MEMS") technology, e.g., Texas Instruments Deformable Mirror Device,[48] and interferrometric devices,[45-47] from Spatial Light Modulators (transmissive and reflective),[41-44] and from parallel optical scanning architectures.[40] New display technologies will result in visually coupled AOI and head mounted display systems that are wide field of view, high resolution, affordable, deployable, compact, and easy to install.

The primary methods of decreasing system latency (other than by increasing update rate and cost) in simulation systems using existing and future display systems will be through the use of new inertial tracking sensor technology, improved predictive filtering, "just-in-time" insertion of sensor data into the graphics pipeline, and image shifting.

TRACKING TECHNOLOGY

One of the most significant contributors to latency in head tracked systems has historically been the tracking sensor itself. Early magnetic head-tracking systems typically contributed 60 ms or more to the system latency. More recent magnetic trackers have been greatly improved, but if low jitter is also required beyond the center of the tracking range, it is necessary to apply some type of low-pass filtering, thus reintroducing significant phase delay. Time-of-flight acoustic trackers are also intrinsically limited in responsiveness (20-50 ms minimum latency) due in part to the slow propagation of sound in air. Optical trackers have no intrinsic requirement for lag, but most commercially available implementations have large latency due to the high computational burden of image processing.

In order to compensate for some of the latency caused by both the tracker and the image generator, inertial sensors have been added to magnetic, optical and mechanical tracking systems to provide head-motion prediction by measuring angular rates and/or linear accelerations directly (instead of deriving them from position sensing), and using them to predict future head pose.[3,16,33] The optical/inertial system at the University of North Carolina (UNC-Chapel Hill) is based on the optical ceiling tracker, an inside-out optical beacon tracking system involving head-mounted cameras aimed at a computer-controlled array of IR LED sources mounted in ceiling panels. Due to the high resolution of the optical position sensitive detector elements, the relatively large size (and higher performance) of the inertial sensors used, and careful calibration and auto-calibration procedures, it achieves superior absolute accuracy suitable for augmented reality applications requiring precise registration with a real-world reference frame.

Inertial sensors are essential components of Inertial Navigation Systems (INS) used in guidance instruments on aircraft, ships, and weapon systems. Miniaturization of the basic INS components (angular rate gyroscopes for orientation and linear accelerometers for position) has led to new uses for inertial sensors, e.g., air bag release sensors in automobiles. Further size and weight reductions resulting from micro-machining (MEMS) technology now permit their use as inexpensive, moderate fidelity sensors for human body and object tracking.[5] Inherent properties of inertial sensors offer the potential for high resolution, low latency, unlimited range/working volume (requires no transmitter/receiver configuration), and immunity from noise, electromagnetic interference (EMI), and line-of-sight obstructions. One or more of these features is lacking in all other existing tracking technologies.

A low latency head tracking system based on inertial sensors alone has been developed, as well.[6-8] This system successfully uses an inclinometer and compass to curtail any long-term drift in orientation derived from the angular rate gyros, and has been packaged into a commercial sourceless *orientation* tracking system (InterSense IS-300). In this system, it is not necessary to first record and average numerous samples in order to filter out noise and produce a stable output. Instead, a single reading can be processed and made available to the application in about 2 milliseconds.

Unfortunately, no method exists to curtail *positional* drift of linear accelerometers in a sourceless tracking system. Therefore, any 6-DOF tracking system that needs to exploit the advantages of inertial sensors for high update rates and improved prediction must be a hybrid system combining inertial sensing with source-based position tracking technology. The above purely inertial tracking system has been combined with an acoustic tracker to produce a 6-DOF system.[5] As in the above cited hybrid systems, extended Kalman filtering is performed to accomplish optimal sensor fusion of the disparate measurements coming from multiple sensor types, however in the case of the InterSense tracker, the inertial sensors are the primary tracking components.[4]

Hybrid tracking systems using inertial sensors achieve latency comparable to that of mechanical trackers, while retaining the range of motion of magnetic, acoustic and optical trackers. The InterSense IS-600 hybrid inertial/acoustic system has somewhat lower accuracy relative to the UNC-Chapel Hill system due to the present limitations of micro-machined inertial sensors, and the higher variability of acoustic sensors compared to optical sensors, however it is commercially available and relatively low-cost.

PREDICTION

An accurate mathematical motion model of the underlying vehicle (or other moving object) dynamics of a simulation can be used to better predict the future state of the simulated moving vehicle. This has been applied with some success to compensate for image generation latency in flight simulators.[15] Because of the well defined nature of flight dynamics (vehicle angular and linear velocities and accelerations do not change very much over an interval of a few frames), it is straightforward to predict vehicle motion well enough to compensate for image generation latency.

By contrast, head-tracked simulations are adversely affected by *end-to-end* latencies as small as 50 ms or less,[2] and head motion dynamics are quite rapid and unpredictable compared to vehicle motions. It is, therefore, very important to use head-motion prediction to reduce the simulator head-motion-to-visual-feedback latency, which includes latency due to tracking, communications, image generation and display scan-out. Non-inertial trackers normally are not able to directly measure motion derivatives (angular and linear velocities, accelerations, etc.), which are necessary for extrapolating position forward in time, so these quantities have often been estimated from successive position data records, sometimes using a Kalman filter[17,18] The high sampling rate of inertial sensors and their direct measurement of motion rates and accelerations greatly improves the effectiveness of motion prediction algorithms, by about a factor of 5 compared to non-inertial predictors.[33]

Unless high-fidelity computer graphics rendering with single-frame latency becomes feasible and cost-effective, further improvements in predictor performance will be required in order to significantly reduce latency in head tracked systems. It will be especially important to reduce latency in mission rehearsal systems in order to minimize the readaptation time required before the operational mission can safely be undertaken without risk of simulator aftereffects.

There are two basic approaches which may lead to significantly better prediction (increased prediction interval and reduced error). The first is model-based prediction and the second is adaptive prediction. All of the predictors described above are model-free, in the sense that no specific model of human head-motion dynamics is used. Instead, the head is assumed to be an unconstrained rigid object with 6 degrees of freedom (DOFs), and kinematic relations between acceleration, velocity and position are assumed to operate independently for translation and rotation. In model-based prediction, human-specific sizes, masses, muscle forces and anatomical constraints are considered.

In one implementation of the model-based approach, the head was simply constrained to move at the end of a neck of length L, pivoting about a fixed point, which resulted in a 20-30% improvement compared to a model-free approach.[11] This is an overly simplistic model that does not account for head mass, neck stiffness, additional degrees of freedom at the base of the head, or limits on the rotation possible about the base of the neck.

At the other extreme, a second model-based system has been proposed based on a complete physiological/ biomechanical model of how nerve impulses to the neck muscles lead to head motion.[12] By using the Kalman filter to estimate muscle activation instead of position and velocity state variables, all the constraints and dynamic equations of the head are incorporated, giving maximum prediction without estimating many redundant highly correlated state variables. The approach has not been implemented, perhaps because it is too complex a model.

A more practical model-based implementation is being developed by Hans Weber as a dissertation at UNC-Chapel Hill.[13,14] By estimating prediction state variables in a body-centered coordinate system (BCCS), Weber expects to take account of anatomical constraints and kinesiological data on joint limits and perhaps motion statistics in a much more tractable way. The goal of the research is to achieve a ten-fold improvement over model-free prediction. Weber's model will allow for motion of head, neck and waist-referenced coordinate frames.

The second approach to improving prediction algorithms is to use adaptive Kalman filtering algorithms. Radar aircraft tracking systems widely employ a technique called interacting multiple model (IMM) Kalman filtering.[22] This technique is based on the realization that aircraft flight can be categorized into several distinct modes, such as straight flight, banking maneuver, turning, etc., which have distinct dynamic models. The IMM algorithm attempts first to identify when the modes switch, and then to shift the estimation model appropriately. It would seem that head-motion may likewise be parsable into several distinct modes, such as fixating on a stationary object, tracking a moving object, walking, shifting gaze, nodding, etc. If this observation is true, it may be possible to apply the IMM KF algorithms without much modification. Furthermore, adaptive Kalman filtering algorithms should work in tandem with model-based prediction to provide even better results.

Finally, improved prediction may be achieved through further research into mathematical formulation and proofs for sensor fusion based on Complementary, Extended, and/or other new Kalman filter techniques.[4,7] Increases in computational power allow real-time computation of higher-order, larger state vector algorithms that are optimized for a given error tolerance in either position or orientation accuracy.

An advanced tracking system will need to provide prediction that is adaptable to the dynamic prediction interval dictated by the latency characteristics of a specific computer architecture (i.e., not only to adjust for latency changes of a new configuration, such as replacement of serial communications with a wireless link, but also *real-time* instantaneous changes due to image processing system overload). The tracking system should also facilitate a direct interface with image generation hardware, and synchronization with image panning techniques. (The need for these features is discussed separately in this paper, below).

JUST-IN-TIME INSERTION OF SENSOR DATA

Azuma and Bishop also point out that the error in motion prediction increases quadratically with the product of prediction interval and rate of motion.[33] Motion bandwidth is an inherent characteristic of a particular task, however prediction interval can be minimized with careful simulator design. In contrast to flight control inputs, there is no need for the host computer to process head tracking data. This data can be processed directly by the image generator to compute a new eye point, and thereby save at least one frame of latency.[23,24]

A direct head tracking interface to the Image Generator was developed by Evans & Sutherland for the Close Combat Tactical Trainer (CCTT) Program in order to eliminate unnecessary latency in the host computer. A direct interface is also being developed by E&S for the new Harmony image generator in support of the NAWCTSD (Navy) Transportable Strike/Assault Mission Rehearsal System (T-STARS), which will use a head tracked AOI projector in a deployable mini-dome display. Synchronizing the initiation of the head tracking sample such that it is ready just before it is needed can also minimize latency from asynchronous processes and data communications.

In multi-stage image generators, an additional frame of latency can be trimmed by performing an over-cull within a slightly enlarged viewing frustrum, and then adjusting ("hot-wiring") the eye point with new sensor data just prior to initiating the draw process. This can be performed with only a handfull of Performer calls on an SGI Onyx Workstation. In support of a project for the Berlin based Heinrich Hertz Institute,[10] Division, Inc. implemented a more sophisticated pre-draw call-back capability for insertion of just-in-time sensor data, which accounts for synchronization of multi-process, multi-pipe applications on SGI Onyx Workstations.

Insertion of sensor data just prior to the draw process reduces the prediction interval and, therefore, prediction accuracy. Even at an update rate of 30 Hz, only 50 milliseconds of latency remains after the cull process in a typical image generator (33.3 ms for draw and 16.7 ms for display refresh), which is within reasonable accuracy limits of current prediction algorithms. Fidelity of (head tracked and non-head tracked) 60 Hz flight simulators can potentially be improved using this approach, because real latency is being reduced as opposed to effective latency using prediction alone. In the case of the non-head tracked flight simulator, viewpoint data generated by the host flight dynamics could be used one frame earlier by inserting it directly into the draw process.

IMAGE SHIFTING

Head tracker based deflection of a rendered 2D image was conceived by Rediffusion in the late 70's, and subsequently implemented at the NAWCTSD Visual Display Research Tool (VDRT).[35-38] A measured end-to-end latency of only 12 milliseconds was achieved at a head frequency of 2 Hz. The system was evaluated favorably in air-to-ground and air-to-air tasks.[38] Other researchers found image deflection to be especially effective for target tracking tasks.[26] Later experiments performed at the NAWCTSD Visual Technology Research Simulator (VTRS) facility demonstrated that control over the pixel location at which raster output is initiated in the image generator frame buffer can achieve equivalent results to deflection of the displayed image.[25] This technique transfers hardware complexity from the display device to the image generator.

Image shifting requires very little image generator hardware complexity, namely the ability to perform a hardware pan at the display refresh rate within a slightly over-computed image in the frame buffer. (The amount of over-computing can be minimized through prediction of the following computed eye point.) Hardware panning has been a common feature of PC standard graphics adapters used by game developers since the 80's to achieve faster response and greater interactivity. Moreover, the software control algorithm needed to determine the amount of vertical and/or horizontal pan that is optimal for pitching and yawing head or vehicle motion is also not complex. If fact, the technique does not require any hardware specific support at all if the video board provides the ability to double buffer and an adequate pixel copy (bit-blt) rate.[27-29]

Accelerator boards have as much to gain from image shifting as high end image generators, because they typically support only single stage rendering, and must drop to a lower update rate to generate high scene complexity images. Primary Image is the only manufacturer of single board PC graphics accelerators that is known to have supported image panning (for use in fixed eye point training systems, such as Air Traffic Control).

In support of this paper, Angus Dorbie of Silicon Graphics, Inc. has demonstrated a limited hardware panning capability on the SGI Onyx Workstation. The panning is based on the SGI Dynamic Video Resolution (DVR) feature. DVR only supports operations in 4 pixel minimum horizontal increments (1 pixel vertical), therefore the panning also has the same restriction. (For a 40 degree horiz. FOV and a 7.5 deg/sec head yaw rate, panning would be smooth at 4 pixels/ frame.) This capability is still considered potentially useful, and will be experimented with in order to determine whether smooth panning can be achieved above a minimum head motion rate threshold. Results will be posted on Mr. Dorbie's web site (www.dorbie.com).

In 1995, Evans & Sutherland introduced an image shifting feature trademarked as Multiple Image Suppression (MIS) into their ESIG line of image generators. The MIS feature was successfully used to eliminate the severe multiple imaging and simulator sickness in the Army's Close Combat Tactical Trainer (CCTT) Commander's Popped Hatch scenes caused by turret slewing and vehicle heading changes in 15 Hz update rate tank and fighting vehicle simulators.[23]

The initial implementation of MIS eliminated multiple imaging caused by pure rotation in yaw and pitch, but not roll of the eye point. However, stepping of objects in the peripheral field of view (beyond about +/-30 degrees) is evident whenever MIS is activated and there is significant translational motion of the eye point. This can be effectively compensated for by limiting the horizontal FOV of each channel to about 60 degrees, and compensating for translational motion on an individual channel basis. Rigorous compensation for eye point roll is complex,[30,31] however prototype MIS experiments demonstrate that computing the required image shift separately for each video channel can reduce the stepping in the periphery field of view caused by translational motion, and to a limited degree from vehicle/aircraft roll as well. Objects that are relatively close to the eye point may still exhibit some stepping, but this could be minimized by accurate prediction leading to only small image shifts each frame.

Adaptive or selective use of MIS is the best solution to take maximum advantage of MIS, and minimize the potential for any adverse effects caused by its use. For example, in 30 Hz low cost trainers, such as the Aviation Reconfigurable Manned Simulator (ARMS), or the Crew Station Mission Equipment Trainer (CSMET), MIS could be disabled during aggressive roll maneuvers, or for low altitude/high translational motion. However, MIS would be enabled for normal flight, and especially during an autorotation maneuver in order to eliminate multiple imaging in the out-the-window scene, sensor displays, and mast mounted sight display when either the helicopter or mast mounted sight is being slewed to lock onto a target.

A recent enhancement to the CCTT MIS implementation provides for adaptive use of multiple image suppression in gun sights. Applying MIS in the 15 Hz update rate gun sight imagery improves the Gunner's ability to search and detect targets while slewing the turret. However, MIS can cause multiple imaging of targets that have specific relative motion with respect to the turret, therefore MIS is disabled under program control if there is a target within the Gunner's field of interest and the angular velocity vector of the target matches the operator's viewing motion. Multiple imaging of gun sight reticles is avoided by rendering them on an overlay card. The E&S Harmony image generator video cards will include built in post-rendering symbology overlay, which can used for displaying clear and stable HUD and reticle imagery with or without MIS activation.

A general concern of image generation systems is temporary computational overload, and its effect on image motion quality. With the appropriate algorithm in place, image shifting can be used to compensate for lost frames when update rate reductions occur. In these periods, the image generator would be programmed to continue image shifting until the system is able to complete the computation of the next image. Experiments indicate that small discontinuity in motion can be seen, but it is minor when compared to the effects of overload without MIS. Moreover, the use of this technique can preserve a constant system latency, even with varying frame times caused by uneven image generator loading. Importantly, a constant lag is easier to adapt to than a varying one, and results in less simulator side effects.

Perspective error distortion caused by image shifting will be minimized if the amount of shifting required each frame is also minimized. This is accomplished through the use of just-in-time insertion of sensor data and accurate prediction. If frame rate fluctuations are excessive, or there are large polygons close to the eye point (such as in a Military Operations in Urban Terrain (MOUT) database, then some distortion may be evident, e.g., parallelogram distortion for yaw motion. With improved prediction and with tracking rates that exceed frame rates, each successive image can be rendered based on a highly accurate predicted eye point, and then shifted by only a small amount each frame prior to display output based on the most recent tracker sample. The result is an image that is correct for the instantaneous head viewing vector with very little latency or distortion.

LATENCY MEASUREMENT

The need to control latency is widely acknowledged, however, rarely accomplished outside the military training system arena, because of the difficulty in verifying it. Defining the latency associated with a given simulator cue has traditionally required the use of a high speed strip chart recorder and (optionally) a signal generator and various motion sensors. While this equipment was capable of defining the end-to-end latency of a simulator cue, it offered little insight as to the specific lag associated with any given system component. For example, the strip chart can't directly detect an inefficient executive or interprocess communication that is introducing unnecessary delays.

The SIMulator Evaluation System (SIMES) was developed under a NAWCTSD Phase II SBIR Program with the former Systems Control Technology group of SAIC, and incorporates a reflective memory interface to the host computer, as well as a repackaged version of the Electronic Visual Display Attitude Sensor (EVDAS) developed by the Wright Laboratories at Wright Patterson Air Force Base. Using these features in conjunction with PC-based data acquisition boards, SIMES provides the user with the ability to track a signal as it propagates through the simulator, and to quickly identify areas where latency may be reduced.

SUMMARY AND CONCLUSIONS

As simulator database complexity and display resolution continue to increase, conventional rules of thumb regarding acceptable levels of latency and image artifacts will no longer apply. This will especially be true of head tracked systems and lower cost systems operating at less than 60 Hz. Moreover, deployment of simulators on moving platforms such as aircraft carriers and other ships will exacerbate aftereffects, and further demand that more aggressive approaches for both reducing latency and improving fidelity are taken.

Advancements in tracking sensors, prediction accuracy, and simulation architecture allow simulation latency to be greatly reduced. When combined with image shifting, the simulator designer is provided with powerful new options, including the ability to: (a) radically reduce effective latency and eliminate multiple imaging in a 30 Hz or lower update rate system without increasing cost, (b) increase the database complexity and display resolution of a system without increasing system latency or cost, or alternatively, (c) decrease the cost of a system without increasing effective system latency, or decreasing database complexity or display resolution.

Using latency reduction techniques, cost/performance can be further optimized by allowing the host computer and image generator to update at different rates. Database reuse is facilitated, because a low-cost part-task trainer can faithfully render and interoperate at a lower update rate over the identical database (with no feature thinning or level of detail range scaling) that is used by a high fidelity simulator. Moreover, highly usable low update rate systems can be fielded, and easily upgraded later with the addition of hardware processing components, as funds become available.

Although the cost of image generation hardware has decreased significantly, there has been a corresponding increase in requirements for synthetic environment complexity, geospecific features, and image quality (equating to far greater numbers of textured polygons and antialiased pixels rendered with advanced shading and lighting capabilities). The cost/performance of new technology is also opening up new simulation based training opportunities where there is not an established culture with standards of minimally acceptable fidelity for effective training. Technologies and techniques now exist that call for a new analysis of requirements and to arrive at systems characteristics that are optimal in terms of fidelity, system costs, and portability for a given application.

The optimal development environment would allow system designers to experiment with various latency reduction techniques for a given application. Image shifting could be used adaptively based on the unique characteristics of the application, e.g., degree of moving model motion, symbology, and eye point roll (pure or inertially coupled). In order to support this approach, image generator manufacturers will need to provide application developers with the appropriate controls and documentation needed to efficiently implement the features presented here. The level of effort required is more than compensated through enhanced application performance and cost savings (as well as making new applications affordable). Through judicious use of improved tracking, prediction, architecture, and image shifting, performance of a 30 Hz (or lower) update rate system can be made acceptable for the majority of training applications.

REFERENCES

Latency and Simulator Sickness/Aftereffects

1. Aftereffects and Sense of Presence in Virtual Environments: Development of an R&D Agenda, *7th Int'l Conf. On Human Computer Interaction (HCI)*, Stanney, et. al., International Journal of Human Computer Interaction, 1997, In Press). Contact Dr. Kay Stanney, STANNEY@iems.engr.ucf.edu for reprints.

2. P. Dizio and J. Lackner, “Circumventing Side Effects of Immersive Virtual Environments,” ’ *Proc. 7th Int'l Conf. On Human Computer Interaction (HCI)*, San Francisco, CA 1997.

Inertial and Hybrid Tracking

3. G. Welch and G. Bishop, “SCAAT: Incremental Tracking with Incomplete Information,” *Proc. SIGGRAPH '97 Conference, In Computer Graphics, ACM Annual Conference Series 1997.* http://www.cs.unc.edu/~welch or ~gb

4. E. Foxlin, “Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter, ” *Proc. IEEE 1996 Virtual Reality Annual International Symposium* (VRAIS '96, Santa Clara, CA Mar30-Apr3, 1996), pp 185-194.

5. E. Foxlin, M. Harrington, and Y. Altshuler, “Miniature 6-DOF Inertial System for Tracking HMD's,” *Proc. SPIE/AeroSense 98 Conference*, Helmet and Head-Mounted Displays III, Orlando, FL, April 13-17.

6. E. Foxlin, M. Harrington, and G. Pfeifer, “Constellation: A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications,” *Proc. SIGGRAPH 98 Conference* (Orlando, Florida, July 19-24, 1998) Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH.

7. E. Foxlin and N. Durlach, “An Inertial Head-Orientation Tracker with Automatic Drift Compensation for Use with HMD's,” *Proc. Virtual Reality Software & Technology (VRST) '94 Conference*, 23-26 Aug., 1994, Singapore, pp 159-173, World Scientific, G. Singh, S. Feiner, D. Thalmann, editors.

8. E. Foxlin, “Inertial Head-Tracking,” M.S. Thesis, Dept of E.E.C.S., MIT, 1993.

9. http://www.cs.unc.edu/~tracker, and http://www.cs.unc.edu/~olano/papers/latency

Just-in-Time Insertion of Sensor Data

10. R. Skerjane and S. Pastoor, “New Generation of 3-D Desktop Computer Interfaces,” The Engineering Reality of Virtual Reality, *Proc. SPIE Electronic Imaging Conference*, 1997. (http://atwww.hhi.de:80/blick/papers/spie97/spie97.html , and http://atwww.hhi.de/~pastoor/

Head Motion Models

11. Y. Akatsuka and G. Bekey, "Compensation for End to End Delays in a VR System, " *Proc. VRAIS 98 Conference*, IEEE Computer Society Press, Atlanta, GA, March 1998, pp 156-159.
akatsuka | bekey@robotics.usc.edu

12. http://duchamp.arc.nasa.gov/~arin/arin.html

13. Predictive Tracking Notes (Motion Models) home page: http://www.cs.unc.edu/~weberh/local.html (Hans Weber)

14. H. Weber, "Predictive Head Tracking Using a Body-Centric Coordinate System," Dissertaion Proposal, April 9, 1997, University of Carolina at Chapel Hill, Computer Science Dept., http://www.cs.unc.edu/~weberh/predict/predtrac.html

Prediction

15. F. Cardullo, Frank, and Y. Brown. Visual System Lag: The Problem, The Cause, The Cure. *Proc. IMAGE V Conference*, (19-22 June 1990, Phoenix, AZ), pp 31-42

16. S. Emura and S. Tachi, "Compensation of Time Lag Between Actual and Virtual Spaces by Multi-Sensor Integration," *Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems* (MFI 94), pp 463-469.

17. M. Friedmann, T. Starner and A. Pentland, "Device Synchronization Using an Optimal Linear Filter," *Proc. Symposium on User Interface Software and Technology*, Hilton Head, SC, 1991.

18. J. Liang, C. Shaw, and M. Green, "On Temporal-Spatial Realism in the Virtual Reality Environment, In *Proc. Symposium on Interactive 3D Graphics*, Cambridge, MA, 1992.

19. Kalman Filter home page: http://www.cs.unc.edu/~welch/kalmanLinks.html

20. Ronald Azuma and Gary Bishop, "A Frequency Domain Analysis of Head-Motion Prediction, " *Proc. SIGGRAPH '95 Conference*, In Computer Graphics, Annual Conference Series 1995, pp 401-408.

21. R. Azuma, E-mail correspondence, http://www.cms.dmu.ac.uk/People/cph/VR/Trackers/lag.refs.html

Adaptive Prediction

22. Y. Bar-Shalom and X. Li, Estimation and Tracking Principles, Techniques, and Software, Artech House, Boston, MA, 1993.

Image Panning/Shifting/Deflection

23. R. Moore, "Multiple Image Suppression, " *Proc. 18th I/ITSEC Conference*, Orlando, FL, Dec. 1996.

24. C. Pope, D. Joseph and D. Seymour, "The Use of Head Tracking Technology in Armored Vehicle Simulation," *Proc. 17th I/ITSEC Conference*, Albuquerque, NM, 1995.

25. B. Riner and B. Browder, "Design Guidelines for a Carrier-Based Training System," *Proc. IMAGE VI Conference* (Scottsdale, AZ 14-17 July, 1992)

26. R. So. and M. Griffin, "Effects of Time Delays on Head Tracking Performance and the Benefits of Lag Compensation by Image Deflection," *Proc. 1991 Conference of the American Institute of Aeronautics and Astronautics (AIAA).*

27. Tomasz Mazuryk et.al., "Zoom Rendering: Improving 3-D Rendering Performance with 2-D Operations," TR-186-2-95-09, July 95 (http://www.cg.tuwien.ac.at/pub)

28. T. Mazuryk et.al., "Two-Step Prediction and Image Deflection for Exact Head-Tracking in Virtual Environments," *Proc. EUROGRAPHICS '95 Conference*, Vol. 14, No. 3, pp 30-41.

29. T. Mazuryk et.al, "High Fidelity for Immersive Displays," Tech. Report: TR-186-2-96-02, Jan 96 (http://www.cg.tuwien.ac.at/pub)

Image Shifting Post Processors

30. W. Mark, L. McMillan and G. Bishop, "Post-Rendering 3D Warping," *Proc. '97 Symposium on Interactive 3D Graphics,* (Providence, RI, Apr27-30, 1997), pp 7-16. http://www.cs.unc.edu/~billmark

31. W. Mark, G. Bishop & L. McMillan, "Post-Rendering Image Warping for Latency Compensation," UNC-CH Computer Science Technical Report #96-020. http://www.cs.unc.edu/~billmark

Augmented Reality

32. R. Azuma, "A Survey of Augmented Reality," In *Presence*, (August 1997), 355-385. (See especially Section 4.3) http://www.cs.unc.edu/ ~azuma

33. R. Azuma and G. Bishop, "Improving Static and Dynamic Registration in an Optical See-through HMD," *Proc. SIGGRAPH '94, In Computer Graphics, Annual Conference Series 1994*, pp 197-204. http://www.cs.unc.edu/~gb

34. M. Jacobs, M. Livingston and A. State, "Managing Latency in Complex Augmented Reality Systems." http://www.cs.unc.edu/~us/Latency/ManagingRelative Latency.html

NAWC-TSD Visual Display Research Tool (VDRT)

35. D. Breglia et al., "Helmet Mounted Laser Projector," *Proc. IMAGE II Conference*, (Williams AFB, 10-12 June 1981), pp 241-257.

36. D. Breglia, "Helmet Mounted Laser Projector," *Proc. 3rd I/ITSEC Conference*, Nov. 1981, pp 8-18.

37. D. Breglia et.al, "Visual Display Research Tool, Performance vs. Design Goals," *Proc. 7th I/ITSEC Confernence*, Dec. 1981.

38. B. Browder, "Evaluation of a Helmet-Mounted Laser Projector Display," *Proc. 1989 SPIE Conference*, (Orlando, FL), pp 85-89.

39. D. Burbridge and P. Murray, "Hardware Improvements to the Helmet Mounted Projector on the Visual Display Research Tool (VDRT) at the Naval Training Systems Center," Helmet-Mounted-Displays, Jerome Carollo, Editor, *Proc. SPIE 1116*, pp 52-60 (1989).

Display Technology

40. R. Bergstedt, C. Fink, G. Flint, D. Hargis, and P. Peppler, "Microlaser-based displays", in Cockpit Displays, Darrel Hopper, Editor, *Proc. SPIE*, Vol. 3057, 362-367 (1997). bergstedt@laserpower.com

41. J. Eichenlaub. "A Unique Photonics System Design that Increases the Resolution of an LCD," Electroluminescent Materials, Devices, and Large Screen Displays. *Proceedings of the SPIE*, Vol. 1910, Bellingham, WA: Society of Photo-Optical Instrumentation Engineers, February 1993.

42. J. Eichenlaub, J. and M. Katafiaz,, "Increased Resolution on an ICFLCD Display through Field-Sequential Subpixel Illumination," *1998 Society of Information Display (SID) Conference*, Digest of Technical Papers, Volume 29, Anaheim, CA: Society for Information Display, May 1998. (mek@dti3d.com)

43. O. Woodard, Sr. and D. Vu, "Development of a 2560x2048 AMLCD for HMD Applications, " *12th SPIE Aerosense Conference*, Orlando, FL, 13-17 April, 1998. ollie_woodard @kopin.com

44. B. Ratna, H. Li, S. Nelson, and P. Bey, Jr, "New Electroclinic Liquid Crystals for Fast Gray-Scale Applications, *1997 Society of Information Displays (SID) Conference*, May 13-15, Boston, MA, pp 207-210.

45. R. Apte, F. Sandejas, W. Banyai and D. Bloom, "Grating Light Valves for High Resolution Displays," Solid State Sensors and Actuators Workshop, June 1994 (see also: http://www.siliconlight.com/frames/press/Pwest.html)

46. http://www.microdisplay.com

47. M. Miles, "A New Reflective FPD Technology Using Interferrometric Modulation," *1997 Society of Information Displays (SID) Conference*, May 13-15, Boston, MA, pp 207-210.

48. T. McDonald and L. Yoder, "Digital Micromirror Devices Make Projection Displays," *Laser Focus World*, August 1997, pp S5-S8. See also: www. ti.com/dlp/doc…ers/mems/2dmdarc.htm#fig5, ti.com/dlp/docs/papers/mems/4fabhtm

AIAA-98-4177

AN INTEGRATED VEHICLE SIMULATION FOR THE DEVELOPMENT AND VALIDATION OF A COMMERCIAL REUSABLE LAUNCH VEHICLE

Seamus T. Tuohy, Ph. D.
Simulation Laboratory
The Charles Stark Draper Laboratory
Cambridge, MA 02139

Abstract

In this paper, an overview of the Integrated Vehicle Simulation (IVS) being developed to support the Kistler Aerospace Co. K-1 reusable launch vehicle will be presented. IVS is a six-degree of freedom, variable mass high-fidelity simulation that will support the entire project development including: GN & C algorithm development, GN & C software development, functional testing and mission validation, and hardware-in-the-loop avionics and vehicle integration and validation. IVS is being developed in the Draper Laboratory Simulation Framework. The Framework provides both a standard set of tools (such as data visibility, plotting, recording, etc.) but also provides a methodology in which flight software can be developed, integrated and tested and then, since the code is in ANSI C, ported directly to the target hardware. In addition, once the code is cross-compiled onto the target platform, the hardware can then be integrated back into the simulation through hardware-in-the-loop interfaces. In this manner, one simulation can support project development from conception though flight testing and into operational deployment.

Introduction

The Kistler Aerospace Corporation (KAC) K-1 launch vehicle is a two stage to orbit, reusable launch vehicle being developed to address the need for commercial, low-Earth orbit satellite launch capability. Draper's role concentrates on GN&C, systems engineering, and avionics integration. The Integrated Vehicle Simulation (IVS) is a high-fidelity simulation for two stage reusable launch vehicles and a facility to support development of vehicle avionics systems. IVS is currently supporting the K-1 launch vehicle development in the following areas:

- **Vehicle Mission and GN&C Conceptual and Preliminary Design -** closed loop dynamic simulation for development, analysis and testing of vehicle performance and GN&C algorithms.
- **Avionics Integration, Test & Evaluation** - closed loop dynamic simulation interfaced with flight hardware through a simulation interface unit (SIU).
- **Integration and Development of GN&C flight code** - development environment in which flight code can be developed, integrated and tested in a closed loop dynamic simulation and then ported, without modification, to the target hardware and operating system.
- **Launch Facility Test** - Closed loop simulation of the vehicle as it responds to prelaunch commands from the ground control station.
- **Field Data Analysis & Evaluation -** comparison of simulation generated data with logged telemetry for model validation and flight test analysis.

The fleet of K-1 vehicles being developed by the KAC will provide routine, cost-effective delivery of payloads, primarily communications satellites, to low Earth orbit. This is to be achieved by developing a fleet of fully reusable space boosters, emphasizing low design and development cost along with operational efficiency. The figure below shows the general layout as well as defines some of the terms to that are referred to later in the paper.

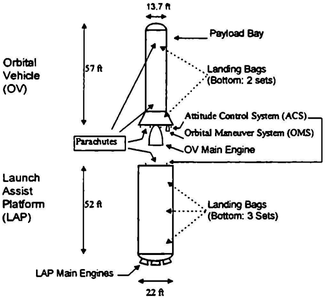

13.7 ft
Payload Bay
Orbital Vehicle (OV)
57 ft
Landing Bags (Bottom: 2 sets)
Attitude Control System (ACS)
Orbital Maneuver System (OMS)
Parachutes
OV Main Engine
Launch Assist Platform (LAP)
52 ft
Landing Bags (Bottom: 3 Sets)
LAP Main Engines
22 ft

Figure 1 K-1 Vehicle Layout

A general mission is shown in Figure 2. The first-stage Launch Assist Platform (LAP) boosts the second-stage Orbital Vehicle (OV) to an altitude from which the OV can reach the desired orbit. The first stage then returns safely to its designated landing site. The OV continues the ascent to the designated low earth orbit where the payloads are deployed, continues on-orbit for a twenty-four hour period, then deorbits, reenters the earth's atmosphere and continues to its landing site. The vehicles land by a combination of parachutes and airbags.

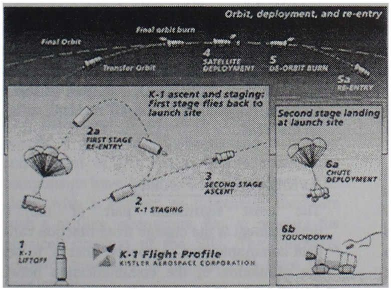


Figure 2 K-1 Mission Profile © 1997 KAC

In this paper, an overview of the IVS is presented. The paper begins with describing the various configurations that the developed simulation will support, proceeds with a description of the development effort for the simulation including vehicle dynamics and models, describes the facilities provided to support software development (GN&C) and the avionics integration, and then ends with a description of the current status of the development effort. It is hoped that the reader will take away an impression of both the magnitude of the support effort as well as a description of how such a large, integrated and complex system development can be aided by the concurrent development of a single simulation.

IVS Configurations

IVS is being developed to support various configurations in regard to development and testing support. This includes various levels of hardware-in-the-loop (HIL) configurations. The completed IVS and HIL Testbed will include the vehicle simulation software, graphics/animation/data visualization, and the simulation interface unit (SIU). IVS is able to execute in real-time with capability to start (and restart) simulation at any phase of the mission with proper initialization. IVS simulates both stages of the K-1 vehicle (LAP and OV) with one vehicle (at a time) being HIL capable (since only for a period of two minutes are the vehicles mated as one, and in this mode, the OV is not very active) but both with full 6DOF vehicle dynamics and environmental. IVS provides the ability for fault injection to test avionics, integrated vehicle performance and procedures. IVS is hosted on an SGI multiprocessor workstation interfaced to a simulation interface unit (SIU) with the option for an additional workstation simulating ground support/mission.

Software Development

IVS is being used to support the K-1 vehicle avionics development and testing capability from vehicle design inception through complete system life cycle. In order to minimize program start up, a parallel evolution of model fidelity and system detail is begin pursued. To support early GN&C system designs and trade studies, IVS rapidly achieved closed-loop with a full set of equations of motion (as described later), first principle subsystem models (developed based on experience from other projects of this type) and with simple, perfect GN&C (note that these were just placeholder and were removed). Therefore, this version encompassed aspects of all systems and mission phases using low levels of system detail. By incorporating lower fidelity models, however, GN&C software development could proceed with placeholders created for more realistic K-1 specific models. This process of updating simulation software, while causing a configuration management task, nevertheless helped GN&C development to commence immediately since at all times there was (and is) a fully functional as-built simulation for the K-1 vehicle. IVS contained representations of all hardware/software systems such as: vehicle mass, aerodynamic, engine performance, gravity, launch, ground reaction, atmospheric, and controller models among others that are described in more detail later in the paper.

We are currently on our third design cycle and upgrade for IVS and throughout the additions and updates, and the development of I/O needed to support the HIL capability, GN&C software development was allowed to proceed (although with necessary incorporation of vehicle model changes). The model and data configuration of IVS has been handled initially by the simulation team, then later by a formalized change review panel with participation from the simulation and software development teams, from the GN&C and program management. IVS will continue to the end of the program to support flight software integration and testing in software only simulation (described as a functional level simulation) through the use of configuration switches that will allow the simulation to communicate directly, without the need of hardware, to the flight software through the established vehicle interface.

VMC Test Facility

Once the IVS *functional level* has been established, the first step towards HIL testing is to establish the interfaces for 1553 communication. Figure 3 shows this configuration of the IVS and HIL Testbed. To integrate the vehicle avionics hardware into the simulation for testing and validation, a Simulation Interface Unit (SIU) has been developed. The SIU is the test control interface using a bus controller simulation hardware board. Since the vehicle mission computer (VMC) is the bus controller for the vehicle, each of the controller simulation models contain remote terminal software. Commercial 1553 interrogation electronics and software are used to verify the 1553 bus with interface electronics and a bus monitor can be used to diagnose anomalous behavior. Through this interface, the 1553 network communications can be tested and validated. In addition, the simulation can use the SIU to interface with hardware which need to be driven by the simulation dynamics through *modeled input* for the systems. One such example would be the navigation sensors.

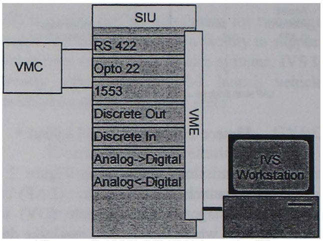


Figure 3: HIL VMC Test Facility

At this stage (just 1553), IVS was ready to support inclusion of the VMC in closed loop HIL testing. The VMC contains all software for GN&C and vehicle subsystem management. These tests will demonstrate inter-process coordination, communication, data flow, fault detection and recovery and will verify integrated vehicle performance and closed loop mission performance (including . pre-launch procedures subsystem checkout, recovery procedures subsystem checkout, powering off, safe vehicle, and data post processing). The goal of this test phase is integration of the all resident software on the VMC.

Avionics Test Facility

The most significant part of the IVS development is the change from function calls to data communication protocols. That is, to begin the simulation of the communication between the hardware devices. Initially, since the simulation is used in the development of flight software, the communication between models are more a matter for coding efficiency. Now, however, the concern is the use of the simulation for hardware integration and testing and validation. Therefore, parallel to the effort of flight software integration is the establishment of I/O and communications requirements using supplied Interface Control Documents (ICDs).

Once the SIU of the previous section has been developed has been developed, any number of controllers can be integrated for test and validation of 1553 data communication. IVS can be used open-loop configuration by sending test messages/signals across the 1553 network to test this ability. The next step is to interface with *the other side* of the hardware. That is, the discrete and analog signals representing command and feedback from the controller to the hardware (which is always simulated either in software or in a combination of hardware/software simulation).

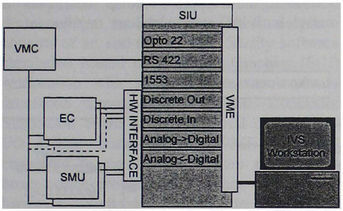


Figure 4:HIL Avionics Test Facility

To prepare for full integration to establish the avionics test facility, a build up of the SIU has occurred with the addition of the needed number of discrete I/O and analog-to-digital, digital-to-analog, and serial interface boards with associated drivers on the workstation to establish communication. The ultimate goal is to establish a facility as shown in Figure 4.

The SIU consists of three 1553 simulation boards, a VME based chassis and a number commercial of I/O boards. These commercial components have been used to construct the SIU along with a number of supporting hardware such a cabling, connectors, etc. to produce the physical structure of the SIU. The following software necessary to run the SIU have been developed:

- SIU native software
- 1553 Simulation Board Driver Software
- SIU I/O Driver Software (software necessary to drive the I/O boards installed in the SIU).

Once the SIU build-up has occurred, then, as hardware is delivered and 1553 communication checked out, the dynamics loop will be closed until, piece by piece, the entire suite of avionics is running with closed loop dynamics.

Ground Control Test Facility

If the procedures of flight readiness, launch and recovery, and ground monitoring are to be validated, then models need to be developed for these procedural and non-avionics systems. This means that the previous set-up is expanded with the additional workstation(s) representing the ground/mission control. This set-up is shown in Figure 5

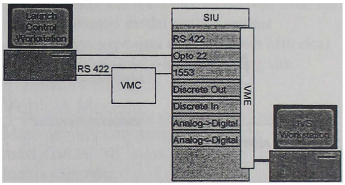


Figure 5: Ground/Mission Control HIL

The functions of the Ground system include the transportation to the launch pad, erection, emplacement of umbilics and fuel lines, loading mission loads and initiating the launch procedure. During prelaunch, communication between each vehicle stage and the mission control computer (MCC) is via an RS-422 serial data connection.

To achieve command and control loop closure around the MCC, models are needed for the fueling operations with I/O for the on-board fuel tanks so that correct feedback can be carried back through the 1553 bus and, eventually, down the RS-422 link to the MCC. Note that some of the models are needed to test the performance of specific hardware that may be in the loop. For example, initialization of the IMU. Others test the ability of the caution and warning system and the system management software.

IVS Overview

IVS is a multi-vehicle, six degree-of-freedom, variable mass simulation. Significant models developed include the following. Multiple aerodynamic models (and logic for seamless transition) for when the vehicles are connected, in each other's wake during separation, or in free air. Mass models to simulated the effect of fuel depletion, payload deployment and actuator movement. Tank models for sloshing dynamics due to the presence of a free-surface as fuel depletes. Propulsion models (which are integrated with the mass models) for both liquid fueled and cold gas systems as well as venting operations. In addition to these, parachute, airbag and sensor models are also developed. All models are complemented by a complete motion parameter computation capability which can report on such things as forces, moments, loads, and position and attitude rates in multiple coordinate reference frames.

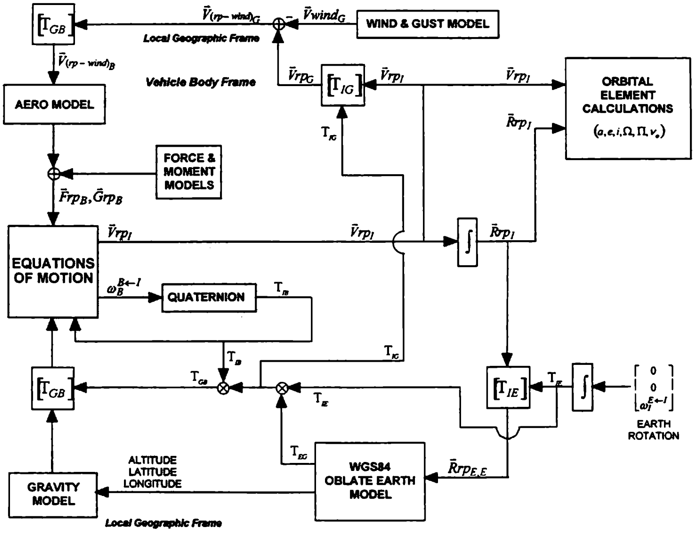


Figure 6:EOM Simulation Loop

IVS provides dynamic environment models in which the full capabilities of both the vehicle and GN&C software can be tested. The environment includes an Earth geode reference (including gravity gradients), atmosphere, wind and turbulence models, vehicle ephemera, and accurate depiction of celestial elements (sun, moon, and stars).

Draper has a long history in simulation development to support system design, development, test and operation. Because of these experiences, Draper has a team of highly qualified, experienced analytical engineers with a proven ability to formulate and solve complex problems including total system real-time simulation with HIL to support all phases of major programs from design through deployment and operations.

Of particular significance is the use of a core simulation methodology that is generically applicable with an inventory of developed, verified, and applied tools, techniques and models with state-of-the-art equipment and facilities. This core simulation methodology centers about the *Draper Simulation Framework* which provides system visibility, animation graphics, plotting and data analysis, simulation management tools, sequencing and interactive control, hardware integration, and test signal generation (for open loop HIL test and verification).

Development Environment

The IVS is being developed in ANSI following a Rapid-Prototyping (or Spiral) Development plan. The structure of IVS is hierarchical and modularized to facilitate hardware integration (through I/O isolation) and to separate functionality for efficient evaluation and testing of flight software code. IVS was initially developed with lower fidelity models so that a complete vehicle system was established early in the program. The models then evolved to higher-fidelity as they became available and to higher level-of-detail as hardware was integrated into the simulation configuration. The final system consists of an end-to-end high fidelity software simulation hosted on a Silicon Graphics (SGI) workstation with simulation driven hardware interfaced through a 1553 Bus Controller Simulation board and commercial off-the-shelf I/O boards installed in a VME rack and connected to simulated hardware of breakout boxes which provide the interface to the flight hardware. Switching inside the simulation software is used to isolate models as they are replaced with hardware.

Equations of Motion

The equation of motion set developed for this program take into account a variable mass, mass propellant, six degree of freedom vehicle. Of particular interests are that for a period of time, two vehicles are mated. In this mode, the propelled vehicle (which is the LAP during ascent) carries the additional static and slosh mass of the OV. The OV, in turn, derives its translational and rotational motion directly from the results of the equations of motion of the LAP. At separation, these values are used at initial states for the OV equations of motion which are then integrated. Figure 6 shows the complete simulation loop where $[T_{AB}]$ implies a coordinate transformation matrix from the A to the B frame.

Environment Models

Environmental models are used to provide data to generate forces and moments and to establish physical constants on the vehicle due to the local environment that the vehicle sees as it flies it's trajectory. The models include:

- WGS84 reference geode used as the Earth reference
- gravity model up to an including J4 term, gravity gradient model which assumes spherical shape of the Earth
- standard US76 and Gram95 atmosphere models to determine local ambient air density, speed of sound, and temperature
- accurate depiction of the celestial elements (sun, moon, and stars) in accordance with simulation time and referenced to an almanac derived orientation of the Earth.
- statistical wind models derived from the Gram95 model truthed against measured data obtained from historical records of the launch/landing site.

Vehicle Models

Mass models are composed of the following mass, center of mass, and mass moment of inertia elements:

- dry (inert, non-moving, non-expendable)
- moving (engine bells and, for the OV, the payload deployment mechanisms consisting of the dome, fairing, elevator)
- propellant by percentage of full tank with slosh
- other expendables (payload and chutes)

The masses are composed form summing the various components to achieve the total mass, center of mass, and mass moment of inertia elements. For the propellant, lookup tables are used where mass is the independent parameter and is computed from the engine mixture ratio and mass flow rate (which is itself a function of the current throttle setting). Slosh dynamics are modeled as harmonic damped oscillators with coefficients derived from a similar lookup table.

Aerodynamic models are based on linear interpolation of coefficients of normal and axial force and for pitch moment as a function of

Mach and angle of attack for the following different flight regimes:

- Separation: Lap in OV wake
- LAP and OV: freestream,
- OV on-orbit
- and stack configuration (OV mated to the LAP during launch and ascent

In addition, for the critical OV reentry flight regime, pitch and yaw damping.

Vehicle Subsystem Models

Main Engines

For the main engine thrust, 1st order, rate limited, lag models are used for normal throttling. In addition, there exist separate start-up and shut-down transient profiles which are derived from test data. During flight, the thrust is also adjusted for ambient pressure (i.e. reduced). The thrust is directed through thrust vector control (TVC) using two actuators for each engine for yaw and pitch control, respectively. These systems are modeled as second order response systems with hysteresis modeled as a transport delay.

For the OMS engines, there are start-up and shut-down transient profiles which derived from test data (there is no throttling capability). For the OMS TVC system, first order response models are used.

For the cold gas attitude control system (ACS), linear ramp functions with start-up and shut down delays are used. Because the cold gas jets are located inside each vehicle cavity (the forward cavity for the LAP, the aft flair for the OV), impingement models are incorporated.

It should be noted that these models are currently used for GN&C development, for final vehicle testing, proprietary models supplied by the hardware vendors, and used exclusively for this program, will be present

Separation

There are four components to the modeling of the separation dynamics. The first component deals with the aforementioned initialization of the OV state vector for the continued (after separation) independent integration of it's equations of motion. The second component is the switch from using the mated aerodynamics to using the freestream, for the OV,, and the LAP in the OV wake (after separation has caused the LAP to maneuver outside of the wake, then the freestream aerodynamics are used also). The third and fourth aspect are the pressurized interstage cavity and the separation bolts.

To propel the two vehicles apart during staging, the interstage cavity maintains positive pressure during ascent. The vehicles are kept mated using bolts. When these bolts are released, staging occurs. To model the separation system, an ideal gas law is applied to produce the resulting force.

One concern during staging is the effect of having a time differential in the mating bolts releasing. This effect is modeled using a two-dimensional dynamics model to produce the resulting angular acceleration for the vehicles. This angular acceleration is used in solving an isolated rotational equation of motion, finding the needed moment for the current vehicle (both OV and LAP) configuration, and inserting this moment into the vehicle equations of motion integration.

Parachutes

For the dynamics of the vehicle while under parachutes, two models are needed. The first is the dynamics of the parachute itself (i.e. increase in drag) and the second is the coupling dynamics of the harness. The harness is what transfers the loads imparted by the parachute onto the vehicle.

For the parachute model, the force is assumed to be parallel to the vehicle velocity vector (but directed in the opposite sense). The force is derived from the familiar equation

$$F = ½ mV^2 CdS$$

where CdS represents the drag coefficient multiplied by the parachute surface area. The values of CdS re obtained from a lookup table keyed to the vehicle state.

Harness models are used to model the application point of the derived force. For the harness models, two types are used, the first models the effect of the total angle of attack of the vehicle on the application point on the vehicle during high speed trajectories. This model consists of rigid links that do not carry compressive forces nor moments at the attachment points (either the vehicle or the parachute risers). For lower speed trajectories, torsional damped harmonic oscillator models are used for pitch and yaw. The vehicle/parachute system will not resist roll motions.

Airbags

Prior to landing, inner and outer airbags are inflated. The outer airbags are intended to decelerate the vehicle, the inner airbags are used to support the vehicle after landing. The vehicle is aligned to its velocity vector to minimize the probability of vehicle roll-over during landing. The outer airbags decelerate the vehicle by providing pressure relief needed to stop descent of the vehicle. They are modeled using ideal gas law to determine the force exerted on the vehicle. The inner bags are modeled as linear damped harmonic oscillators.

Satellite Deployment

After the OV achieves the correct orbit and orientation, the payload bay nose dome is unlocked, opened and articulated off to one side. Then the payload fairing is retracted and the payload elevator is extended.

At this point the payload is fully exposed such that deployment can occur either longitudinally or laterally. The payload is then released by activating the deployment mechanism. After the payload is ejected, the payload elevator is retracted and the nose dome is closed and locked. The payload fairing is left in a retracted state for landing.

The articulation models for the dome, fairing and elevator are all modeled as linear systems. The payloads are ejected using an impulse force applied at the COM of each satellite with a reaction force applied to the vehicle. As was the case during staging, the current state vector (with the addition of the velocity from the impulse force) are used as initial values to integrate a separate equation of motion set for each satellite. The equations of motion for the satellites consist of only the effect of the separation transient and gravity. Full orbital elements are derived for each satellite to confirm deployment and to determine separation (i.e. possible collision and engine burn safety zones).

Sensors

Sensors that are important for GN&C development are modeling the space integrated IMU/GPS (or SIGI) system for the OV, a star tracker for the OV, a stand alone inertial navigation system for the LAP, and TVC actuator position feeedback sensors. For the SIGI and IMU, the vendor supplying the device has provided models of the navigation equations (i.e. strapdown) and the navigation filter. Models for the performance of the IMU and GPS systems are being derived from specifications.

Finally, data downlink and uplink for the vehicle is being handled by a TDRSS system. In IVS, line of sight from the TDRSS antennae on the vehicle to the TDRSS satellites is evaluated.

Vehicle I/O Models

I/O models for the 1553 data communication are being developed based on the program 1553 ICDs. These models will include methods for packing and unpacking of 1553 messages, routing information, and the messages themselves. I/O models needed to close the loop on the hardware side are being developed based on individual contractor interface documents and wiring lists. These models will include methods for constructing appropriate messages and for the timing and routing of these messages for various hardware that will be on the vehicle but will be simulated by IVS. In addition, a software interface for these models to the software drivers of the I/O boards will be developed. These interfaces will be tested against contractor supplied data for moding command and response I/O (based on a checklist, for example) and test results and data.

Animation

Dimensionally accurate 3D models of the K-1 vehicle: OV, LAP, parachutes , airbags, satellite deployment mechanism, a textured earth model (for on-orbit scenes), as well as a star chart have been developed in OpenGL for visualization of the complete K-1 mission profile. Models will also include: launch site, launch pad, recovery site, and earth limb. Visualization allows for viewing of flight phases from many perspectives, e.g. ground observers, “chase plane”, OV, or LAP.

IVS Development Environment

CVS, a commercial source code control software package, is being used for source code traceability and development stability during development. A version control process for the working vehicle simulation has been established to enable configurations IVS to be maintained. That is, to support the rapid-prototyping model of development, versioning of models are coordinated as they are integrated into the vehicle simulation. This facilitates the process of configuring specific installations for use in the support of testing and development of the K-1 vehicle avionics. Version control is maintained using the *Draper Simulation Framework* standard Makefile set utilizing the *GnuMake* utility. Versioning is maintained through the establishment of releases, configuration input files, and through documentation

Progress to Date

We are ready to begin the third round of model upgrades beyond the initial closed loop capability. GN&C code development has progressed to a fully integrated (across mission phase) state. We have been running closed loop with the VMC running a full set of flight code and 1553 communication. We have also started to integrate controllers with discrete I/O capability. We have had open loop tests of the I/O and expect closed loop performance in June. At that time we will start tests with the analog type signals. We have completed the SIU buildup and all drivers needed in the simulation to drive the COTS I/O boards in the VME rack. We are currently writing the I/O models needed to test the functionality of the controllers and resident software.

AIAA-98-4178

AN ADAPTABLE AVIONICS TESTBED FOR REUSABLE LAUNCH VEHICLES

William C. Wagner, Project Manager / Simulation Systems
Elizabeth S. Yingling, Simulation Systems Engineer
Boeing North American, Reusable Space Systems (RSS) Division
Downey, California

ABSTRACT

A distinctive real-time simulation testbed was developed over several different reusable launch vehicle projects, that can be used from early flight software development through flight test. Three different stages during vehicle development evolution require testbed support: initial flight software (FSW) check out with a simulated vehicle, flight software check out with the actual vehicle in the loop, and ground system monitoring during flight test. Since the design-to-flight-test time is typically short, a testbed that could aid a program through all three stages with no stage-to-stage redevelopment was required. The approach to accomplish this objective was four-faceted: the first element was a simulation architecture which allowed for the gradual substitution of hardware elements into the testbed as those elements became available to carry the testbed from an all math model environment to a hardware-in-the-loop environment. Second, a simple but powerful shared memory debugging system was devised to provide visibility into the actual flight software code for both the simulated vehicle and vehicle-in-the-loop configurations, greatly enhancing check out. Third, a ground telemetry and range monitoring system was developed that utilized the same testbed hardware, enabling the simulation to close the loop with the flight vehicle at the test site with no additional hardware. The co-located simulation allowed for on-site trouble-shooting during non-flight test hours, a capability that proved invaluable. Fourth, in order for the user to make a seamless transition from initial flight software check out through flight test monitoring support, reusable 2-D ground system displays were developed and integrated up front.

In order to quickly assemble the testbed, a previously developed Boeing North American (BNA) standard simulation and ground system infrastructure was used to provide basic system functionality such as a simulation executive, 3-D visualization, data collect and display services, all with graphical user interfaces. The simulation software unique to a particular reusable launch vehicle is integrated within this infrastructure and built to run on multiple platforms: first on workstation to support early development and then on a VME-based embedded processor system as avionics hardware elements are added. Various topics are covered as the approach to the testbed design is discussed; these include the simulation architecture (including computing hardware), testbed software design approach, hardware-in-the-loop configurations and ground monitoring system development. Various figures depict the testbed architecture and show the role of the testbed during vehicle flight software development and flight test. The advantages and versatility realized through using this testbed are discussed in detail; topics include, but are not limited to, the advantages of the shared memory system and the co-location of the simulation at the flight test site, and the versatility of the 2-D and 3-D displays through all stages of the project. The simulation testbed described herein represents a distinctive capability in having a single testbed support multiple stages of a design-to-flight-test vehicle development program.

SUMMARY

Boeing North American (BNA) has participated in several reusable launch vehicle projects. In today's accelerated development climate, these programs often require a very short design-to-flight test time. In order to support such a short cycle, a single simulation testbed which could support the program phases from design through flight test was developed. One of the major objectives of the testbed for these programs was to support each phase without major rework or redesign of the testbed. In addition to keeping costs low, program schedules had to be met. A "grow with the program" approach was used, heavily leveraging existing testbed technologies. The testbed software is built in a modular fashion so that hardware can be integrated into the loop as it becomes available. In this approach, the testbed functions first as an all math model simulation; the second stage integrates a commercial equivalent of the vehicle's flight processor (FP) in the loop to allow for early testing of the flight software and concurrent development of communications to the FP itself. After successful testing of the flight software functionality, the FP flight unit is substituted for its commercial equivalent. An integrated GPS/INS navigation sensor is typically also included in the avionics testing. After these elements have been verified, the vehicle itself is added into the loop; this allows the avionics to be installed on the vehicle and be driven, through vehicle wiring, with the simulation in the testbed acting as the environment. This testing takes place prior to the flight test portion of a program. During the flight test phase, the testbed moves out to the flight test site with the vehicle, where the testbed has two functions: one is to be the ground monitoring system for the actual flight tests and the second is to continue to use the testbed for additional flight software checkout and testing in between flight tests, thereby saving both schedule and money.

BACKGROUND

Recently, Boeing North American has been working on a variety of programs involving reusable vehicles that perform launch, on-orbit operations, and re-entry and landing functions. These programs have gone through different stages of development, from paper studies to actual preliminary flight testing, but they all had several requirements in common: short development-to-test schedules, heavy emphasis on state-of-the-art avionics practices - stressing automation, redundancy and testing, fast prototype development spirals to refine the design and flush out integration issues, and early involvement of software-in-the-loop and avionics in-the-loop testing to drive out hidden integration problems and mitigate risk issues early. To support these programs, a common open-architecture approach to flight processor selection and flight software design methodology was adopted. It stresses the use of commercial products whenever possible -- at a minimum, the use of "commercial equivalent" products in early development and laboratory testing. Additionally, the flight software architecture is based on a COTS operating system, making the software as transportable as possible between platforms. Languages, however, have varied based on program specific issues, but have been limited to C and Ada.

Both single- and multi-string avionics architectures have been utilized, depending on the program's design issues. For the purposes of this paper, a prototype vehicle that adopted a single-string avionics system will be used as the example (it simplifies the discussion while still highlighting the extent of the capabilities). The avionics in this "typical" BNA built prototype demonstration vehicle consists of a single-string FP/sensor/effector system. Applying the open-architecture approach to the flight computer and software, it was decided to select a VME-based PowerPC. The Radstone 603e Power PC Flight Processor in a Ruggedized VME Chassis was chosen and came integrated with Wind River's VxWork's Real-time Operating System. The flight box is an 8-slot chassis populated with RS-232 and RS-422 serial ports, several channels of 12-bit Analog-to-Digital (A/D) and 12-bit Digital-to-Analog (D/A) interfaces, input and output channels of Digital/Digital (D/D) signals, and a multi-channel ARINC-429 Interface. There were only three sensors required to support the autonomous GN&C functions: BNA's COTS C-MIGITS Integrated INS/GPS Navigation Unit, a commercial radar altimeter, and an air data system. The vehicle's aerosurfaces are controlled by 4 Electro-Mechanical actuators (EMA) and a nose wheel steering actuator for rollout control. A PCM telemetry unit and uplink and downlink transmitters/receivers are also included.

TESTBED REQUIREMENTS

From the start, the testbed architecture was developed to meet both the requirements of a systems integration lab (SIL) for vehicle integration/checkout and a ground station for the flight test program; the testbed's capability evolved as the needs of the program matured.

During the development phase of the program, the testbed had to support two levels of testing: 1) flight software and hardware integration and

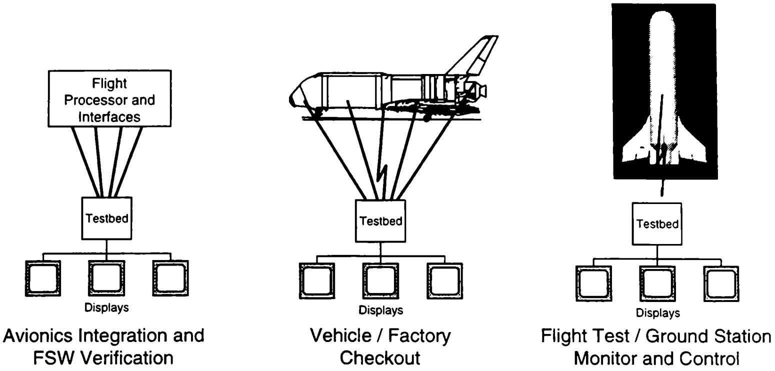

Flight Processor and Interfaces
Testbed
Displays
Avionics Integration and FSW Verification
Testbed
Displays
Vehicle / Factory Checkout
Testbed
Displays
Flight Test / Ground Station Monitor and Control

Figure 1. Testbed Supports All Phases Of The Program

subsequent verification testing and 2) vehicle integration and checkout testing. During the flight test phase of the program, the testbed had to also act as the flight test ground station performing monitoring, control and data archival functions (Figure 1). Although there was no requirement to do so, the testbed was designed so it could also perform in the simulation capacity at the flight test site. This ability to do both functions proved invaluable: anomalies which occurred during flight tests could be debugged, changes could be implemented and then checked out, using the simulation, prior to the next flight - all at the flight test site. This saved not only time, but the effort associated with having to ship data and software back and forth between the flight test site and the development site. Since the domain experts are often at the flight test site, it is the optimum location for all testing.

TESTBED ARCHITECTURE

The testbed's architecture utilized three key elements to enable satisfying the requirements while deploying the testbed quickly: 1) a reusable testbed software infrastructure, 2) a proven flexible/expandable testbed hardware system, and 3) incorporation of an existing off-line analysis vehicle simulation program as the core vehicle dynamics and environment software.

The real-time simulation group at the Reusable Space Systems Division (RSS) of Boeing North American (BNA) has developed a simulation/system infrastructure (S/SI) over the past 11 years which provides a complete range of software tools and services necessary for building, debugging and executing simulations with distributed data display, archival and post-processing capability. It is currently in use on over 10 projects, most of them hardware-in-the-loop (HWIL) testbeds. In addition, the Shuttle's Vehicle Automated Checkout System (VAC) at the Boeing Palmdale facility and the GPS-2F Satellite Checkout Facility utilize the S/SI.

The S/SI is composed of three main parts: 1) a copyrighted and patented simulation framework called SIMEXEC, 2) a data distribution, display and management function, and 3) a collection of support tools and libraries.

- SIMEXEC provides an environment to load, initialize, mode, control, synchronize and interact with discrete-event and/or continuous (cyclic) simulations in either real or non-real time. The architecture is based on a distributed, layered software model, and is currently hosted on several Unix workstations and embedded (VME-based) systems. Migration of user's code between platforms generally only requires recompiling and linking. It is written in C and supports simulation code written in C, C++, FORTRAN, and Ada. SIMEXEC's use of symbol table information allows the user access to all (static) simulation variables. This powerful feature enables initializing, displaying and/or collecting any variable through user-defined lists specified at runtime with no special user code required.

- The distributed data infrastructure provides a variety of tools (e.g., real-time data injection, real-time data displays and archival) to allow the user to interact with the simulation and its data streams. It also contains utilities to access, synchronize, decommutate, archive and display telemetry data. It too provides a common environment ported across several platforms and is written in C++
- Additionally, a set of software libraries is available that provide a large variety of math routines/macros, memory management functions, external interface device-driver libraries, automated scenario control (failure inject scripting), and 3-D visualization (out-the-window or god's-eye-view).

The testbed's hardware system is based on proven elements that have been used in many HWIL and ground system testbeds at BNA. As stated earlier, SIMEXEC has been ported to a variety of host platforms and this capability was utilized in each testbed's evolution: Initial simulation development was done on a Silicon Graphics workstation to take advantage of the easier access to and better debugging capabilities of a workstation compared with those of a VME-based embedded environment. Even the external interface checkout and initial FP integration was accomplished using a VME-based SGI Challenge system as the testbed simulation host processor, bus-extended to a VME-chassis containing the interface boards. This allowed rapid, concurrent and efficient development. Once initial testing was completed, the SGI host processor was replaced with a Heurikon Baja R4700 embedded VME processor running the VxWorks real-time operating system. This change provided a faster processor/operating system, enabling more deterministic real-time interactions. Since SIMEXEC was already ported to this environment, initial rehosting of the already tested simulation took only a matter of days. This approach provided a complete high-fidelity HWIL simulation capability that fit in the space of a 20-slot VME chassis. A Sun Sparc Ultra 170 workstation was also used for user interaction and data archiving. Although this approach is state-of-the-practice today, it is one that BNA has been utilizing for over 9 years (Figure 2).

Once the real-time hardware and software infrastructure was in place to jump-start the testbed development, the simulation models of the vehicle, its subsystems, and operational environment were needed. BNA's GN&C designers modified their off-line analysis program, Shuttle Descent Analysis Package (SDAP), to develop the guidance and

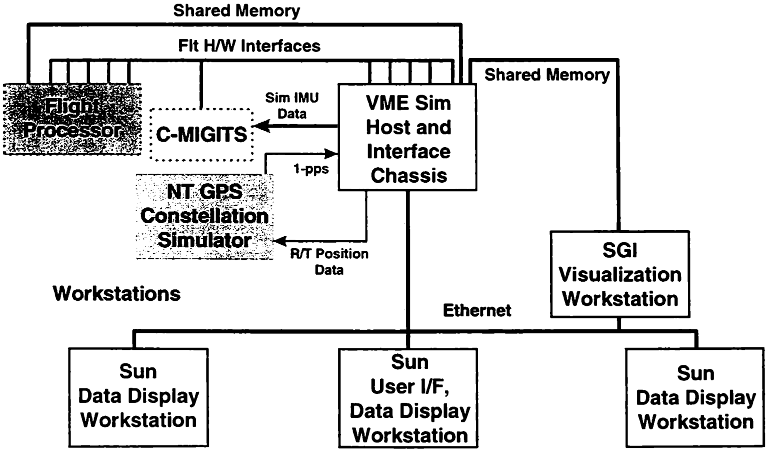

Shared Memory
Flt H/W Interfaces
Shared Memory
Flight Processor
C-MIGITS
Sim IMU Data
1-pps
VME Sim Host and Interface Chassis
NT GPS Constellation Simulator
R/T Position Data
Workstations
Ethernet
SGI Visualization Workstation
Sun Data Display Workstation
Sun User I/F, Data Display Workstation
Sun Data Display Workstation

Figure 2. Testbed Hardware Diagram

control laws for the vehicle. SDAP has been in use for over 20 years to support a variety of programs (not just the Space Shuttle) and it's high fidelity 6 degree-of-freedom (DOF) dynamics simulation is configured for the specific vehicle's airframe. It also includes a fairly complete modeling of the sensor and effector subsystems. It was a fairly straightforward task to rehost SDAP from its batch-type processing environment into the real-time clocked interactive SIMEXEC framework, with the initial attempt being accomplished in about a week. Most of the rehost issues involved compiler differences and ultimately the process was automated so that subsequent updates are completed in a matter of hours. The GN&C designers have been using either Matrix-X or Matlab to develop the GN&C algorithms, with the autocoder feature of those tools producing software to be integrated into SDAP for closed loop analysis testing. The plan was to then reuse this tested GN&C software in the FP. Using SDAP as the simulation software in the testbed made the entire design and testing process very cost- and schedule-effective and seamless. There were several modifications that had to be made to the original SDAP sequencing to accommodate the requirements of real-time HWIL interactions, and these changes then became the baseline for the analysis mode as well.

With SDAP providing the core simulation functions, the major task of building up the testbed was integrating SDAP with the required external device drivers and adding software switching functions to either use internal (SDAP) GN&C commands to close-the-loop or do that through the FP. This switching was easily accomplished by inserting a single subroutine call into SDAP's main sequence code. An imu model that accurately modeled the C-MIGIT's Digital Quartz IRU (DQI) was also added. The functional block diagram that defines the elements in the testbed and their interaction with the FP is shown in Figure 3.

SYNCHRONIZATION, MODING AND TIMING

Two of the most important issues to address early in any testbed development is how the simulation and flight avionics are to be synchronized, and then to insure the data flowing between them represents the correct latency that will be experienced in the real system. This is a crucial step in providing a valid test environment from which to predict performance.

The major subsystem that provides synchronous data to the FP is the C-MIGITS GPS/INS system. It is designed to generate multiple RS-232 integrated navigation and rate

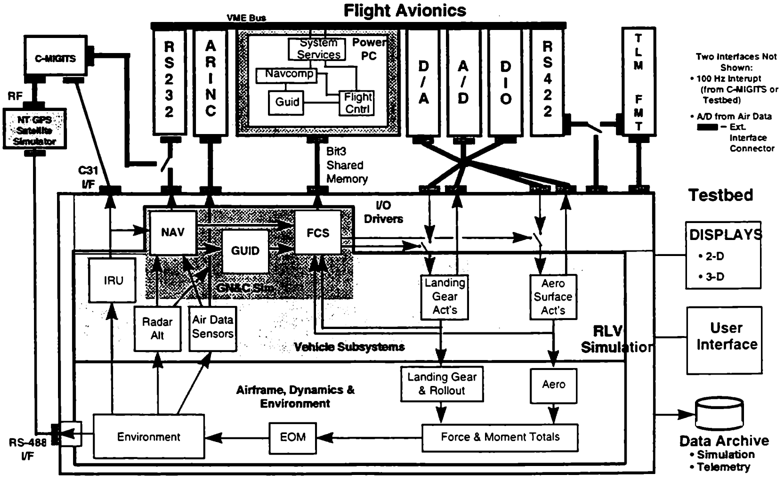


Figure 3. Functional Block Diagram And Interfaces

messages at several frequencies, all synchronized to a C-MIGITS 100 Hz RS-422 "interrupt". Therefore it is mandatory that the cyclic scheduling of the FP and testbed be synchronized with this "clock". Since the C-MIGITS wouldn't be available for initial testing, it was required that its functions be emulated by the testbed; therefore in this mode, the testbed would be the source of the 100 Hz interrupt and resulting RS-232 messages. The C-MIGITS would be in-the-loop at some time during the testing, so a method that was compatible with all expected configurations was necessary. A common reoccurring testbed issue is how to close-loop test an integrated GPS/INS system with the FP without physically moving the IRU (even if a flight table is available, it is impossible to produce the proper translational accelerations). Fortunately, the C-MIGITS is designed to be "test-friendly" and contains several features that make it easy to integrate and control. The 100 Hz interrupt and 100 Hz IRU rate message are essentially the result of six internal 600 Hz DQI messages. There is a special test feature that allows a simulated 100 Hz IRU message to be sent to the C-MIGITS through a dedicated port. When a certain pin is shorted, this simulated IRU signal is used instead of the DQI's and is in turn output in the 100 Hz rate message (to the FP) as well as being used internally to produce the GPS/IRU Kalman filtered navigation solution. In this mode, the 100 Hz interrupt (to the FP) is required to be generated by the testbed at the same time the simulated IRU message is sent to the C-MIGITS. The testbed generated the simulated IRU signal using a PENTEK SBC, since it contains a Texas Instruments C-30 processor that inherently generates a native C31 serial stream format required by the C-MIGITS.

The design solution used is depicted in Figure 4. A Force 40 single board computer (SBC) was used as a C-MIGITS message/interrupt generator. This was chosen because the 115Kbps RS-232 messages are cpu-intensive and it was inefficient to handle them on the simulation host processor (Baja R4700). An Bancom PC03V IRIG-B timecode reader provides a 100 Hz VME-backplane interrupt (that is synchronized from 1Hz timecode signals generated by a Bancom PC05V IRIG-B generator). The Force 40 SBC fields the interrupt and then immediately sends a "mailbox" interrupt to the Baja and an RS-422 interrupt to the FP. The Force 40 then delays 2.5 msec before sending out the RS-232 messages to emulate the actual timing as measured from an actual C-

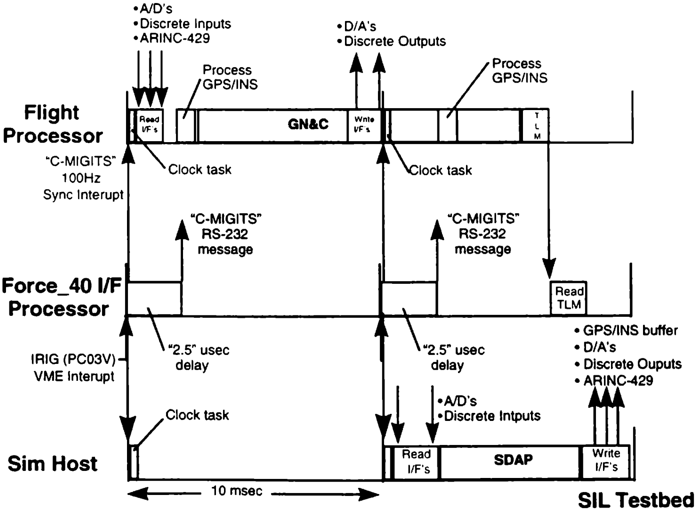


Figure 4. Synchronization and I/O Timing

MIGITS. When the C-MIGITS is in-the-loop, the Force 40 sends an additional mailbox-type interrupt to the Pentek SBC to cause it to send the simulated IRU signal, and the RS-232 message transmissions (from the Force 40) are inhibited. Thus, all time-critical signals are synchronized across the multiple elements.

It was desired to test as much hardware-in-the-loop as possible at some stage of the testing in order to minimize any flight test "surprises". Therefore it was necessary that the real C-MIGITS be tested in-the-loop. To do that, it's GPS receiver and IRU had to be properly "stimulated" in order for it to output messages coordinated with the simulated flight. The method of replacing the IRU was just described. However the system didn't navigate very well without the GPS updates and a Northern Telecon (NT) GPS Constellation Simulator was integrated into the testbed to provide the GPS receiver with a set of simulated RF signals representative of those received by the moving vehicle. The NT required the dynamic state vector data from the simulation via 10 Hz RS-488 messages. The greatest challenge encountered in integrating the simulation, NT, C-MIGITS and FP was in properly synchronizing and coordinating all the messages, mode changes and simulation startup.

FLIGHT PROCESSOR INTEGRATION TESTING

The integration and testing of the FP and it's input/output (I/O) subsystem with the flight software (FSW) was accomplished over many incremental tests using the testbed. Initially a non-ruggedized "commercial equivalent" unit from Radstone was utilized. Later as the ruggedized unit became available, it was swapped in - releasing the commercial equivalent unit for FSW debugging and ultimately for use as a second SIL.

One of the most invaluable testing tools introduced was the ability of the testbed to read/write the FP's memory without any involvement of the FP. This was easily accomplished because the simulation system and FP were VME bus-based. A COTS VME-bus windowed repeater shared memory interface board set was used to connect both chassis' 32-bit address space (no other signals were passed). The boards were configured so that the memory in the FP's PowerPC board was "visible" from the testbed's chassis. The FSW configured all of the data structures that passed between the external interfaces and the GN&C, all the internal structures that passed between the GN&C routines, and all of the FSW modes and states structures to be subsets of one large structure. The FSW then specified the local memory address at which this structure was to be located. For FSW written in C, the ".h" file that defined this structure was used directly to the testbed, along with its FP address. For FSW written in Ada, the ".h" file provided a template used to create an equivalent Ada spec package, creating an extra step. The testbed, through SIMEXEC's memory mapping function, was able to overlay this structure onto the memory pointer and hence created a shared memory area of all the internal FSW data (Figure 5). The only "intrusion" of reading/writing the FP data is the very minimal impact resulting from the "cycle-stealing" of the PowerPC's internal memory bus. It should be noted that a reflective memory solution (such as a VMIC RM or ScramNet interface) would not be as non-intrusive as this approach, since it would have required the shared memory structure be mapped into a memory area outside of the PowerPC - resulting in slower memory access times and a configuration-altered FP.

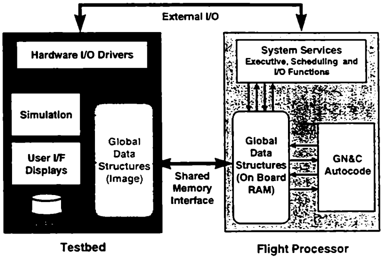


Figure 5. Flight Processor Memory Access Diagram

The benefits provided by this shared memory "tool" was a major advantage to the testing process. Historically, the only visibility that the FSW tester has (once the code is downloaded into the FP) is through telemetry and possibly a monitor/debugger. This "shared memory window" into the FP provides the powerful ability to data collect any (or all) variables in this shared memory. Then on-line or post-run plots can be used to troubleshoot any anomalies. This proved invaluable many times in quickly pinpointing problems in minutes that otherwise may have taken hours or days to analyze.

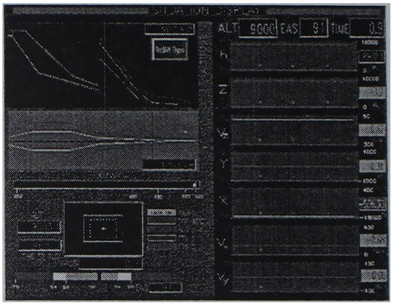
This capability was also used early in the testing process to support an incremental buildup of capability in the FSW:

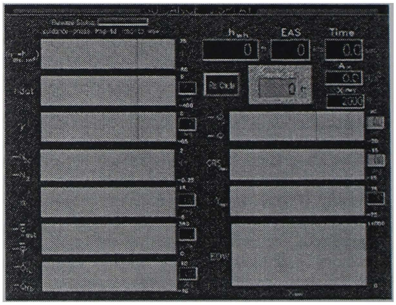
- Once the basic FSW executive and scheduler were written and standalone tested, the testbed was used to monitor shared memory data to test (and debug) these functions.
- Next the I/O drivers (RS-232, A/D, D/A, D/D's and ARINC-429) were added and the testbed sent dynamic data to the FP's external interfaces, stored data into the GN&C command section of the FP shared memory (which was then sent to the output device drivers), and then read the external signals sent from the FP. This series of communications tests quickly verified the signal paths between the external interfaces and GN&C algorithms.
- As the autocoded GN&C software was integrated into the FSW, anomalies arising during testing were quickly resolved: the already verified GN&C software executing within the simulation was run in a "shadow" mode (its output was not used to close-the-loop) with the FSW's GN&C calculations, data was collected from both the simulated and FSW GN&C variables, and post-run comparisons were immediately made that quickly pinpointed the problem area(s).

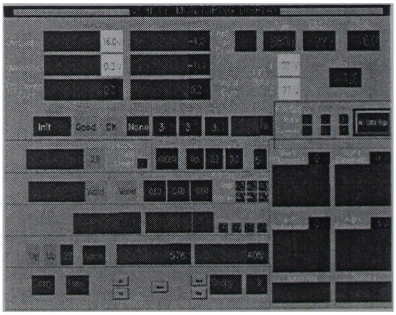
Once the basic functionality of closing the loop (through external I/O) with the simulation was established, the remainder of the testing focused on refining the FSW algorithms and adding uplink and telemetry. One of the features that was built into the FSW was the ability of the testbed (through shared memory) to command the FP through its various states and modes. In this way the testbed could control the entire system, and therefore tests could be cycled through quickly, automatically and repeatably.

In order to quickly assess the performance of the tests, some on-line visualization of the data was required. The S/SI provided two different capabilities:

- Dynamic data from the simulation and/or FP was displayed in real-time to one or more of 5 different 2-D data display pages, selectable by the analyst. The pages were built using the DataViews graphics display tool (DVdraw) and dynamically connected to data streams from the testbed via the Graphics Display Tool (GDT) capability of the S/SI. The data streams were created at runtime by specifying a list file within the GDT. The number of display pages up at any one time was limited only by the number of terminals (X-Window capable) available. (Figure 6 contains several examples.)
- 3-D viewing of the simulated vehicle was easily accomplished by supplying terrain and







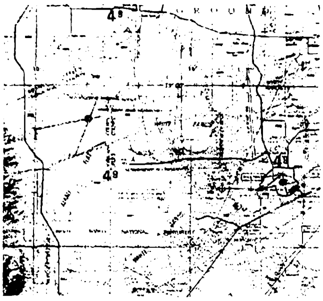


Figure 6. Testbed On-Line Display Examples

(CAD/CAM derived) vehicle 3-D databases to an S/SI visualization tool that connects to the simulation via a few user embedded function calls. In addition to the basic vehicle, articulated parts (aerosurfaces, landing gear, etc.) can also be moved based on the simulation/FP data. A god's-eye- or set of vehicle mounted camera views can be user selected to monitor the flight.

GROUND MONITORING SYSTEM

Once the basic closed-loop functioning of the FSW was established, the telemetry and uplink functions needed testing. The distributed data elements of the S/SI were employed to quickly build the ground monitoring system. A database for the vehicle-specific telemetry was generated and along with a Berg 4422 interface board was used to decommutate and convert the telemetry stream from the avionics PCM unit. A Motorola MVME167 SBC was used to execute the decommutation software and transfer the raw data to a Sun workstation (via a ScramNet reflective memory board) for archival and on-line displays. In addition, the range monitored vehicle state-vector "truth" data via the installation of a special unit mounted in the vehicle. This unit transmits independent GPS/INS-based data to receivers at the range where errors are reduced via differential corrections at their ground station. This data is made available to the BNA ground station via a 10 Hz Ethernet message. This "truth" data is then on-line correlated with the telemetry data and is also available for archival and on-line displays. Since the simulation testbed and ground monitoring system are both based on the S/SI, the 2-D display pages built earlier were used, without change, with the telemetry data - the only change was to the list file specifying where the data was coming from: simulation/FSW versus range/telemetry. (See Figure 7.) The ability to play back the archived data through the displays was also inherent in the capabilities of the S/SI.

Also, since this was the first time this function was required, a small amount of additional software was written to further connect the telemetry/range data stream outputs to the 3-D visualization system. In this way the range "truth" and perceived telemetry navigation state could be simultaneously viewed - displaying, in an intuitive format, their relationship. This is similar to having a view from a "virtual" chase plane available.

In looking at the hardware required for the ground station in Figure 7, notice that, with the addition of the external hardware interface boards and related cabling, all the elements are there for providing a HWIL simulation capability. This feature is being used at the flight test site to close-the-loop with the vehicle in the hangar in between flight tests. This is a powerful capability for troubleshooting anomalies and testing FSW changes with the test team immediately after flight tests.

VEHICLE INTEGRATION AND FLIGHT TESTING

Once the avionics, which were supporting testing in the SIL, were ready to be integrated into the vehicle, the testbed was moved to the "factory" assembly area to support vehicle integration testing. The vehicle wiring was designed so that certain signals could be routed to/from the testbed, either to replace a vehicle subsystem or "listen in". Initially the testbed was used to monitor the open-loop subsystem integration tests (air data, actuators, radar altimeter, C-MIGITS, brakes, telemetry, etc.). It ultimately provided the ability to "fly" the integrated vehicle to test its mission performance.

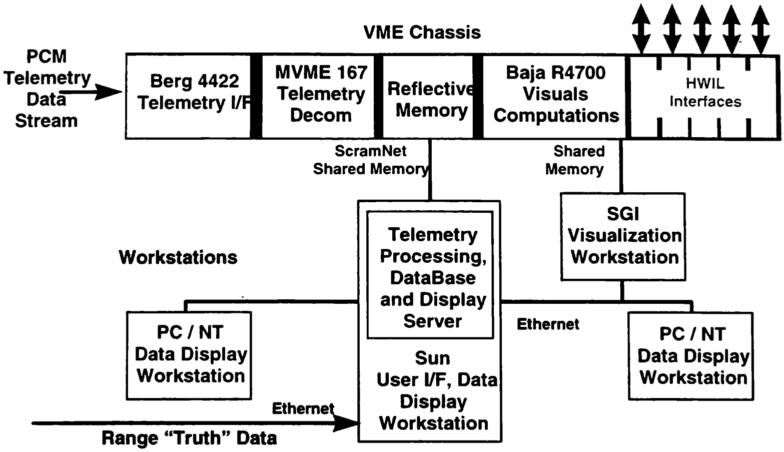

VME Chassis
PCM Telemetry Data Stream
Berg 4422 Telemetry I/F
MVME 167 Telemetry Decom
Reflective Memory
Baja R4700 Visuals Computations
HWIL Interfaces
ScramNet Shared Memory
Shared Memory
Workstations
Telemetry Processing, DataBase and Display Server
Sun User I/F, Data Display Workstation
SGI Visualization Workstation
Ethernet
PC / NT Data Display Workstation
PC / NT Data Display Workstation
Ethernet
Range "Truth" Data

Figure 7. Ground Monitoring Station Hardware Diagram

Once the flight test program began, the testbed was moved along with the vehicle out to the flight test site. There, the testbed/ground system's VME chassis and CPUs from the associated workstation and PCs used for data display were mounted in a rack inside a van. This set up provided a mobile testbed; during flight tests, it was driven to a site next to the airfield runway, hooked up to a commercial power source and used as the flight test ground monitoring station receiving data through telemetry. When flight tests were not being conducted, the van was parked in a hangar where the vehicle was housed; the VME chassis could then be cabled to the vehicle as required and additional trouble shooting and testing could be done with the vehicle in the loop.

CONCLUSION

A simulation-based testbed was developed that successfully supports the full range of testing for a reusable launch vehicle program's development/test cycle, from: GN&C design to FSW testing, avionics integration, vehicle integration and checkout, and ultimately through flight testing. Adopting modular, rapid prototyping testing methodologies, it's configuration evolves with the program, efficiently supporting testing at each stage without redevelopment.

Although its application to reusable launch vehicles was highlighted in this paper, it can (and does) support practically any type of vehicle -- provided the proper simulation software is used.

AIAA-98-4179

The Platform-Independent Aircraft Simulation Environment at Manned Flight Simulator

James Nichols+, Thomas J. Magyar*
Naval Air Warfare Center Aircraft Division
Patuxent River, Maryland

Eric C. Schug**
SAIC / Simulation and Research Services Division
California, Maryland

Abstract

The Manned Flight Simulator (MFS) was created to provide high fidelity simulation capability and support for the Navy's fleet of aircraft. The facility contains five simulation stations, which share a common interface to the computer networks, visual image generators and actual aircraft flight hardware. Any cockpit at MFS can be used in these simulation stations using a "roll-in, roll-out" concept to easily transfer a cockpit from one simulation station to the next. With this interface, a simulation executive was required to run the multiple simulations at any simulation station. The Controls Analysis and Simulation Test Loop Environment (CASTLE) was developed to meet this requirement. Since this initial requirement, CASTLE has greatly expanded and includes many tools for use during a real-time piloted session or for desktop development and analysis use. With new development, popularity and increased performance of computers, CASTLE now operates on a variety of platforms and operating systems using the same source code and graphical user interface (GUI). These operating systems include SGI-UNIX, HP-UX, DEC VMS and Windows95/NT.

Background

The Manned Flight Simulator (MFS) is one facility within the Air Combat Environment Test And Evaluation Facility (ACETEF) at the Naval Air Warfare Center Aircraft Division (NAWCAD) in Patuxent River, Maryland. The ACETEF complex is a collection of laboratories that are used to emulate the complete battlefield environment, including both friendly and hostile elements. These laboratories (see Figure 1) are often connected to various other facilities and test ranges around the world for joint exercises, and may also be operated separately for local testing. MFS provides the high fidelity, piloted airframe simulations for such tests.

MFS is also used to directly support flight test, control law development, avionics development and integration, and new aircraft technology. Both rotary wing and high-performance fixed wing aircraft are modeled.

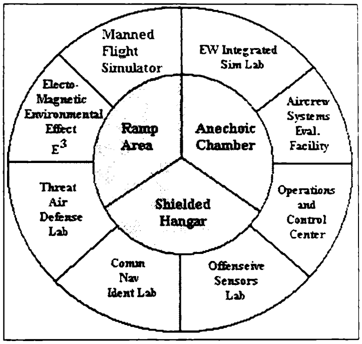


Figure 1: ACETEF

The simulation stations within MFS include a motion-capable platform, a 40ft diameter dome, a helmet-mounted virtual display and two other engineering development stations. The simulation stations share a common interface to the computer networks, visual image generators and actual aircraft flight hardware enabling them to be switched among the simulation stations. MILSTD1553 busses are used to interface mission computers and flight control computers with the lab.

Each simulation cockpit was developed with a "roll-in, roll-out" capability with common structural footprints and electrical interfaces. This allows each cockpit to be transferred from one simulation station to another in a matter of minutes along with its actual flight hardware and desired visual system and processors. Cockpits that reside at the facility include the F-14, F-18A/D, F-18E/F.

[*] Aerospace Engineer, Member AIAA
[**] Aerospace Engineer
[+] Aerospace Engineer, Senior Member AIAA

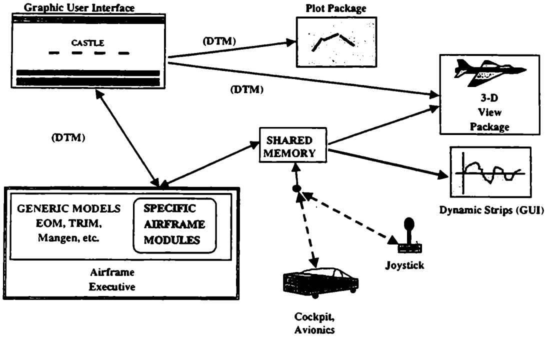


Figure 2: XCASTLE Design

V-22FSD, V-22EMD, and a Multi-Reconfigurable Cockpit (MRC) that can accommodate the throttle, collective, and stick of most aircraft.

The multiple simulations, cockpits and simulation stations required a "generic" open-architecture simulation executive that could run all of the existing and future hi-fidelity simulations at the facility. The Controls Analysis and Simulation Test Loop Environment (CASTLE) was developed to meet these requirements. CASTLE can interface with the labs network of computers and control any real-time (piloted) simulation, or be used at the engineer's desktop for development and analysis with its built-in tools.

First Generation Architecture

The first generation of the CASTLE simulation executive ran solely under the VMS operating system on DIGITAL VAX computers using the FORTRAN programming language. However, the software was created with a modular design so it could grow with the changing technology. It is still in use today and supports the premier Navy aircraft including the F-18 and V-22 in their development phase and F-14 life cycle support. This version of CASTLE has been used for thousands of hours of piloted simulation sessions annually, and for uncounted hours of desktop engineering analysis.

Next Generation Design Goals

Changing technologies in computer operating systems and increased user requirements made it necessary to update CASTLE to use industry-standard features. The most important criteria was to write code that would be platform-independent, using one code set for multiple platforms and computer operating systems. This was accomplished by using the standard programming languages as appropriate. FORTRAN77 and FORTRAN90 were used for the math model of the airframes, and C/C++ was used for the CASTLE architecture and user interface. Another important consideration was retaining all desirable functionality of the earlier CASTLE versions.

A graphical user interface environment was built into CASTLE using the standard X-Windows/Motif user interface tools. This gives the software a familiar user-friendly interface environment and also allows non-intrusive operations during simulation sessions. This GUI code is independent of the airframe executive and is executed as a separate task from the airframe. It can also execute distributed tasks and can accept drop-in tools for expanded capability. This GUI-based software environment is now referred to as XCASTLE, although references to CASTLE are considered to be synonymous.

Overview Of XCASTLE Design

The design of the XCASTLE software is very modular on several levels. It consists of a collection of tasks running simultaneously, with the XCASTLE GUI being the controlling entity. Other processes include the actual airframe executive, plotting packages, a spatial visualization tool, interfaces to the piloted laboratory environment, and various other utilities. These tasks communicate via several methods, which include shared memory and a TCP/IP protocol. Figure 2 describes some of these tasks and the communications methodology. It should be noted that both TCP/IP and shared memory communications can be utilized by any particular process. The TCP/IP protocol used is the Data Transfer Mechanism (DTM) designed by the University of Illinois National Center for Supercomputing Applications (NCSA), and is hosted on a variety of computer platforms. The use of this protocol allows the XCASTLE tasks to be running on separate machines. Data word format conversions of the information packets transferred between these tasks are performed automatically. The shared memory can be resident on one machine for tasks running solely on that machine, or can consist of "global" shared memory such as ScramNet. The shared memory read/write interface will also perform all necessary data word format conversions. These conversions are an important consideration since the large varieties of computers used rarely have the same physical data representation.

XCASTLE GUI Description

The design of the graphical user-interface of CASTLE consists of re-usable C++ classes built on top of the X-Windows/Motif user interface. All the CASTLE facilities are built on base classes that handle commonly performed tasks such as defining the GUI, loading and saving settings and sending information via DTM to the appropriate sub-process. New facilities can be quickly generated by using these base classes. The use of settings files enables the user to restore previous sessions and to save the current session settings. A scripting language is available for performing automated tasks and is also used to perform "macro" operations, which is quite useful for running the jobs in the background.

Besides acting as the master process, the XCASTLE GUI allows the user to interactively define aircraft initial conditions settings, create data flow between airframe variables and cockpit controls, determine data storage requirements, and control the various analysis tools built into the XCASTLE system. Digital data storage and plotting is also controlled from the GUI. A Data Dictionary is available that allows the user non-intrusive access to all documented variables in the current airframe database. Variable values can be modified during a simulation run without suspending operations.

Design of the Airframe Executive

The airframe executive consists of a "shell" of common (or "generic") modules, and a collection of airframe-specific modules linked into a software image referred to as the airframe simulation. The shell code is split roughly into two parts, the first being the looping architecture executed during airframe motion propagation, and the second being an extensive analysis tool set. The majority of this shell code is written in ANSI-C, with the equations of motion and related software still in FORTRAN77. The airframe developer can replace any XCASTLE generic module with a customized version, although this is not a recommended practice for configuration control reasons.

The specific modules that describe a particular airframe are supplied by the airframe developer, and may be written in any language. The most common language is FORTRAN, although several models hosted at MFS are in ANSI-C and Ada. The airframe executive was designed to be flexible enough to rapidly accommodate widely varying model structures. The airframe developer may choose to convert the model into a standard format that can then be operated on by the various XCASTLE development tools, or to merely encapsulate an unmodified model with interface routines. The preferred method is to reformat the model as this allows the user full access to all of the XCASTLE variable tools and provides self-documentation.

One of the most powerful features of the XCASTLE airframe executive is the ability to define the loop architecture interactively. This allows linking in development versions of code and switching back and forth between development and baseline code without re-linking the executive. It also allows the user to create multi-loop frames, and have positive control over execution order. A different loop structure can be used for the run mode, trim mode, and other analysis modes.

The core of the XCASTLE structure is the variable database management system. It consists of a graphic-based editing tool (EDITCOM) and a pre-compiler (CASCOMP). Once the developer has defined a variable database, the pre-compiler is used to generate equivalence statements (FORTRAN) or macro definitions (C/C++) in the target source code. A special assignment routine is generated during the link procedure, which is used to equate the ASCII-based variable database with the address of the variable in the software. This provides a flexible, robust, completely device-independent method of variable access.

Another development tool available with XCASTLE is the Function Table Processor, which is used to convert tabular data into executable source code. This software can interpret function data in many different industry formats, and uses linear interpolation to return function values.

XCASTLE Analysis Tools

XCASTLE encompasses many tools to perform engineering analysis on a simulation during both piloted and non-piloted sessions. It should be stressed that the exact same code set is used for the cockpit-in-the-loop piloted sessions and for non-realtime engineering sessions so analysis results are valid for either mode. New tools can be added easily due to the modular design of XCASTLE.

The data storage capability of XCASTLE saves the values of selected variables during a simulation run into dynamically allocated internal memory. Multiple data sets can be stored at varying rates and output to files in pre-defined formats, including ASCII text column format, Matlab file formats, and user-defined formats. Any variable available through the variable access database in the simulation can be saved, and as many variables as

desired can be saved during a run. This is only limited by the memory constraints of the host computer.
The Maneuver Generator (ManGen) is used to overdrive variables in the simulation. The data used to drive these variables can come from several sources. Several pre-defined functions are available to the user in the Maneuver Generator, including step inputs, ramp inputs, varying sinusoidal inputs, and doublets of these functions. Time history data from flight test or other simulation runs can be loaded into XCASTLE and used to drive any variable during a simulation run. The user can also interactively define a function by creating curves with the plotting package. Multiple functions can be summed into one variable driver.

The Trimming facility uses a one-sided perturbation of each user-defined control until the defined state derivatives have reached a target value. The user can set control limits and error tolerances on the target values. The algorithm is robust enough to allow under-controlled sets as long as the uncontrolled state derivatives are already within tolerance. The simulation can be trimmed to straight and level flight, steady climb/descend and specified angle of attack conditions. The user can also select non-zero target values for the state derivatives and "trim" to accelerated conditions. Non-standard trim controls, such as engine thrust, can be used to artificially trim the aircraft to normally unobtainable conditions in order to extract linear models and perform similar tasks.

The Linear Model Extraction (LME) facility allows the user to define a linear model structure by specifying the states, state derivatives, inputs and outputs. The user can designate which airframe modules are called, and in what order they are executed. The Linear Model Extraction facility will perturb the states and inputs with user-defined perturbation values and obtain the best approximation of the partial derivative for each matrix element in the A, B, C and D matrices for the state-space equations:

$$\dot{x} = [A]x + [B]u$$
$$y = [C]x + [D]u$$

The outputs can be saved to an ASCII text file or saved as a MATLAB file for further analysis. This tool has proved to be invaluable, as it operates directly on the same airframe code set that is used for piloted sessions and control law development in their natural element. Several airframe projects have made extensive use of this feature with excellent results.

The Sweep facility allows the user to exercise the model to look at trends and discontinuities in the model database. The user specifies a range of variable values and increments as inputs and also specifies output variables. The simulation will run through these inputs and display the outputs for each. The output can be saved to a file, or loaded back into XCASTLE for plotting and further analysis. When used in conjunction with the scripting facility, the sweep facility is a powerful tool that can be used to automate repeated tasks using different parameters for each iteration. This can be especially valuable for creating a "trim" map of the flight envelope, or extracting linear models at various points throughout the envelope.

The Real-Time Processing (RTPS) Link is used at MFS to drive a simulation and its visual systems with flight test data during an actual flight test. This is useful for providing parameters that are difficult or impossible to compute at the RTPS facility, and for simulation validation. The real-time link can also drive engineering test stations (ETS) with data from a piloted simulation at MFS for ETS display development and configuration or for pilot familiarization, test envelope expansion and flight test engineering training.

The Integrated Data Evaluation and Analysis (IDEAS) is a separate package that is designed to enhance productivity in the area of advanced flight test data analysis and model development. It is an open architecture system that includes comprehensive database management tools, various System Identification tools, and other user-supplied analysis tools. The primary focus of IDEAS is to rapidly utilize flight test data to identify and improve simulation performance, but the versatility of IDEAS lends itself to many other applications.

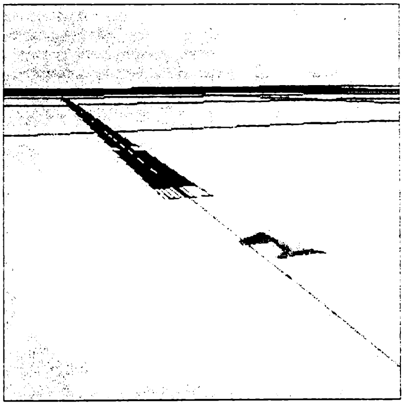


Figure 3: CasTools Chase View

CasTools is a graphics software package that was developed to visualize aircraft trajectories and attitudes in a 3D virtual world. This visual aid may be attached to any external data source, but is normally tightly coupled to a specific XCASTLE airframe simulation session. The CasTools flight display uses non-textured OpenGL graphics for speed and portability. The open architecture of the XCASTLE environment allows more detailed commercial products to be substituted if they are available. MFS uses CasTools as a low-cost desktop "sanity check" before moving the simulation into the real-time laboratory with its high quality visual displays.

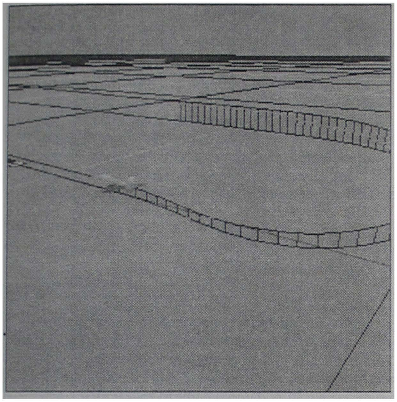


Figure 4: CasTools Flight Path Markers

An unlimited number of viewing positions and orientations can be defined and saved by the user. These options include a pilot view, a fixed viewpoint, and a moving viewpoint attached to the aircraft model. Figure 3 depicts a "Chase" or "Wingman" view of an aircraft on final approach to landing, and Figure 4 shows the same approach from a fixed viewpoint with the flight path trace and altitude bars enabled. Other features include a Heads-Up Display (HUD), a runway, wingtip ribbons and trails, flight path and ground trajectory paths, and altitude bars. In addition to viewing the simulation run in a passive mode, the simulation can be flown with PC-style joysticks, rudder pedals and throttle controls using the built-in joystick capability that exists in the SGI and Windows NT versions of XCASTLE. Trajectory history data files can be loaded into CasTools for viewing and multiple data sets can be viewed simultaneously. Using CasTools, simulation variables can be plotted during a real-time run on dynamic strip-charts that update as the simulation is running. Multiple data sets and multiple strip-charts can be added to view an unlimited number of variables.

Computer Platforms

CASTLE was developed initially on Digital Equipment Corporation (DEC) computers using the VMS operating system. Due to external customer demand and the popularity and increased performance of other computer and operating systems, CASTLE was completely re-designed to run on various computer systems. By using ANSI-standard and POSIX-compliant source code CASTLE is available on DEC Alpha computers running the VMS or Windows NT operating system, Silicon Graphics Inc..(SGI) workstations and Hewlett Packard computers running the UNIX operating system, and Pentium class PC's running Windows95 or Windows NT Workstation. XCASTLE is currently being developed for use on the Apple Macintosh operating system. The Motif X-Windows system was chosen for the Graphic User Interface (GUI) because of its wide usage on various computer systems. The current XCASTLE software is identical for all of the above platforms. The specific computer system or a third-party vendor provides low-level code, such as X-Windows.

XCASTLE Users

XCASTLE was originally developed primarily for use within the MFS facility to support piloted simulation sessions, as well as for engineering analysis and research and development in the desktop environment. The diverse and extensive requirements that MFS is responsible for have ensured that XCASTLE has evolved into a versatile tool with many users and applications. Over the years, XCASTLE has been used to support a large number of US Navy flight test programs, develop and validate control laws, assist accident investigations, and provide an invaluable tool to the warfare simulation evaluators. MFS has provided aircraft simulations to many external users as well, including many universities, contractors, US government agencies, and foreign governments. XCASTLE has supplied piloted aircraft simulations during the Joint Theater Missile Defense (JTMD) exercises and the Joint Combat Search and Rescue (JCSAR) exercises, where a high-fidelity aerodynamic model was required. Several US Marine deployable flight trainers utilize the XCASTLE software, and it is being installed on several other military flight trainers. The XCASTLE software supports inter-player operability through the use of the High Level Architecture (HLA) protocol.

Conclusion

XCASTLE is a very powerful flexible simulation executive that contains over 10 years of requested capabilities and lessons learned. It is capable of running the many airframe simulations at Manned Flight Simulator, from simplistic linear models to the most complex aircraft. The ability to rapidly incorporate external airframe models has proven to be invaluable. Using the exact same code for piloted simulation sessions and at the desktop greatly reduces the development and life-cycle costs of airframe simulations at MFS. XCASTLE has proven to be easily adaptable to external operating environments. XCASTLE contains standard coding practices that enable it to operate on multiple platforms and operating systems. Its modular design allows an unlimited number of tools and applications to be integrated into XCASTLE. With this vast capability and proven technology at Manned Flight Simulator, XCASTLE is an excellent candidate for use as a standard simulation executive for Navy trainer simulations, and across branches of the Armed Forces.

References

[1] Nichols, J.H., "The Generic Simulation Executive At Manned Flight Simulator", AIAA-94-3429-CP, 1994.

[2] Burton, R.A., Miller, C.C., Mills, R.E., "Manned Flight Simulator and the Impact on Navy Weapons Systems Acquisition", AIAA-94-3420-CP, 1994.

AIAA-98-4180

MODIFICATION OF A SURPLUS NAVY 2F122A A-6E OFT FOR FLIGHT DYNAMICS RESEARCH AND INSTRUCTION

Kevin R. Scalera[*] and Wayne C. Durham[†]
Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0203

Abstract

This paper describes the adaptation of a Navy Operational Flight Trainer (OFT) to a university flight dynamics instruction and research platform. The requirements of the modified system are described and compared with the deficiencies of the system in its original state. Completed and future modifications of the trainer to satisfy the requirements are detailed. Representative uses of the flight simulator for classroom instruction as well as research are described.

Introduction

Virginia Polytechnic Institute and State University's Aerospace and Ocean Engineering Department conducts research in several areas of flight dynamics and control funded by Naval Air Systems Command and NASA, Langley. Much of this research deals with issues of implementation of the results of theoretical research into piloted flight vehicles. Research efforts include the practical implementation of non-linear control law design for highly maneuverable aircraft; and research into the problems of redundant aircraft control effectors, the robustness of nonlinear control law design, loss of control, control failure identification and reconfiguration, and pilot-induced-oscillations.

The philosophy under which this research is performed is that fundamental results should proceed to implemented design through stages of simulation with progressive levels of realism and fidelity. The currently followed research methodology originates with the generation of an idea, proceeds to a desktop batch simulation followed by a real-time desktop simulation with pilot input through a 2-axis joystick. The last test of the research concept is a manned real-time implementation. Since the cost of actual flight time can be overwhelming for a university, high-fidelity, piloted, motion-based simulator serves as an viable alternative.

In March of 1996 Virginia Tech acquired the motion-based 2F122A Operational Flight Trainer (OFT) from NAS Oceana in Virginia Beach. This trainer is a fully functional two-seat flight simulator with a three-window visual display and a three degree-of-freedom cantilevered motion system. Figure 1 shows the layout of the simulator. The simulator was originally used by the U.S. Navy for A-6E Intruder pilot training. The simulator has a reputation among fleet pilots for its ultra-realistic representation of the A-6E aircraft and has been particularly valuable for nighttime carrier landing training.

Within the Aerospace Department at Virginia Tech, the trainer will serve two primary purposes: (1) instruction of aircraft flying qualities, and of flight dynamics and control characteristics, and (2) research into aircraft automatic flight control problems.

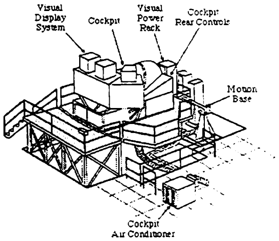


Figure 1: Simulator Layout

[*] Graduate Student, Department of Aerospace and Ocean Engineering. Student Member AIAA.
[†] Associate Professor, Department of Aerospace and Ocean Engineering. Senior Member AIAA.

Requirements, Deficiencies, and Modifications

To serve as an instructional and research tool, the trainer is required to simulate dynamic response characteristics of aircraft other than the now retired A-6E. No attempt is made to simulate specific aircraft subsystems and cockpit features. The focus is wholly on reproducing dynamic and aerodynamic characteristics, and control system mechanization. The cockpit will necessarily resemble that of an A-6E, but with respect to primary flight and engine instruments it will be generic. For instructional purposes the F-16 and F-18 HARV (High Angle-of-attack Research Vehicle), as well as a generic variable-stability aircraft model are utilized. For research purposes the F-18 HARV and the F-15 Advanced Control Technology for Integrated Vehicles (ACTIVE) aircraft are utilized. Future research will require simulation of the F/A-18E/F and Joint Strike Fighter (JSF) configurations to be determined.

Simulation Environment

The original 2F122A simulation environment was necessarily rigid and not amenable to user modifications. Because current and proposed research involves tactical Navy aircraft, compatibility with the architecture of the Naval Air Warfare Center's Manned Flight Simulator (MFS) was desired. Their architecture is called CASTLE (Control Analysis & Simulation Test Loop Environment). CASTLE provides the desired flexibility and analysis tools. Most of the software and hardware modification choices described below were driven by the aim of compatibility with CASTLE.

Generic Math Model

In order to achieve multiple aircraft capability, the trainer must simulate generic equations of motion (math model). The generic math model is to remain independent of the type of aircraft being simulated. A rigid body aircraft operating in a standard atmosphere is assumed. Aircraft-specific subsystems not essential to flight (e.g., radios, fuel transfer, etc.) are not required. Data provided to the generic math model consist of initial conditions, mass and geometry properties, and net aerodynamic and propulsive forces at and moments about the center of gravity, as functions of aircraft state. The user will provide subroutines necessary to interface with the generic math model.

The math model incorporated in the 2F122A is aircraft independent. However, the aerodynamic and engine databases as well as their interpolation routines which utilize these databases are not in CASTLE-compatible format. In addition, the 2F122A subroutines' computational frame length was hard-coded into much of the simulation software. As a result of this structure, the present simulator configuration is restricted to operate at 30 Hz, the rate dictated by the integration time steps.

Modern flight control computers operate at rates that are much higher than the 2F122A's 30 Hz (80-100 Hz is typical). In order to run the flight control computer simulations and the basic simulation at the same rate, the 2F122A integration routines are being modified to be independent of CPU speed and to run on a higher speed host computer. A Silicon Graphics Origin 2000™ Deskside System (180 MHz 2 CPU, 256 MB, 4 GB disk) has been acquired for this purpose. The Origin 2000™ communicates with the Gould 32/67 through a SCRAMNet® Network VMEbus host interface shared memory system. All FORTRAN based simulation code is being relocated on the Origin 2000™ host while the assembly language I/O interface subroutines remain on the Gould 32/67 in order to preserve the present Real Time Peripheral (RTP) interface between the host computer and the visual and cockpit subsystems. It is planned to replace the RTP interface by a more modern VME bus architecture to allow the use of off-the-shelf digital interface boards for most of the communications between the cockpit and the host computers.

Aerodynamic Database

The requirement is to simulate the dynamic responses of various aircraft. The 2F122A was dedicated to simulating the A-6E characteristics. When utilizing a math model which is independent of aircraft, the main characteristics which distinguish one aircraft's simulation from another are the engine and aerodynamic databases and the interpolation subroutines which are specific to those databases. Consistent with the CASTLE architecture, the user will provide engine and aerodynamic databases for simulation. Each engine and aerodynamic database is called by interpolating subroutines that are specific to that database. Any coordinate system transformations required for compatibility with generic aircraft equations of motion will be incorporated into the interpolating subroutines by the user.

Control Loader

Flight dynamics and control research necessarily involves flying qualities evaluations, which are greatly influenced by the control system mechanization. This requires modeling the control stick characteristics of various aircraft. The simulator in its acquired configuration utilized a hydraulically powered control stick and rudder pedal artificial feel system. The characteristics of this system were matched to that of the A-6E aircraft and could not be modified.

A Fokker ECoL 2500 electric control loader system was purchased and installed. The ECoL 2500 is a programmable control loading system which affords researchers the real-time capability to vary the control stick dynamic characteristics. A custom modification of the ECoL 2500 system was ordered to enable rate limiting of the stick for future stick logic (smart stick) research. Communication with the ECoL 2500 occurs over the 8-line Asynchronous serial communications port of the Gould 32/67. Serial communications drivers were written at Virginia Tech and integrated into the existing simulation code.

Although the ECoL 2500 system provides most of the flexibility required for the intended research thrusts, it's flight control model is an unchangeable 2-mass model system. An upgrade to Fokker's ECoL 5000 system is planned, which will enable modification of the flight control model as well as digital control of the simulator hydraulic motion system. In addition, the ECoL 5000 will enable researchers to implement real-time position limiting of the control stick, a requirement for the active stick logic research.

Parameter Recording

It is desired that the trainer system be able to record relevant flight parameters. Post-flight analysis requires time histories of various parameters. Parameters will be selectable by the user. As many as 40 parameters for up to two minutes of simulated flight time, will be required. These parameters include simulated aircraft accelerations, velocities and position variables, simulated control positions, pilot control inputs, control stick dynamic characteristics, as well as simulated control generated forces and moments.

The original configuration of the 2F122A permitted limited recording of variables essential to playback of the simulation. Although this system did allow some parameter recording, it did not allow the flexibility for the user to select the desired variables nor output them into a desired format.

The UNIX-based version of CASTLE (XCASTLE) simulation environment was developed with these concepts in mind. XCASTLE affords the user the capability to select the parameters he desires to evaluate and record them for an unlimited length of time. In addition, XCASTLE contains tools for real-time strip-charting of recorded parameters. Since methods of eliminating unnecessary control surface saturation are being researched, the ability to record and display parameters real-time serves as an invaluable research tool. This allows the researcher to monitor the control law's output real-time and determine which areas of the control law design need increased robustness.

Engineering Workstation

It is desirable for researchers to be able to perform real-time monitoring of research-related parameters and post-flight analysis of recorded data from the cockpit. This permits rapid evaluation of the attainment of simulation objectives. The original configuration of the 2F122A required all simulation control through a low level graphical interface at an instructor's station. The interface did not allow the user any flexibility in modifications of system parameters. In addition, the instructor's station was not equipped with any post-flight analysis software/tools.

To achieve real-time monitoring and immediate post-flight analysis, the Bombardier/Navigator's (B/N) crew station of the cockpit was replaced by a touch-screen engineering workstation that duplicates the XCASTLE environment. This engineering workstation gives Virginia Tech researchers the capability to record relevant flight parameters from the cockpit and perform analysis utilizing software such as MATLAB[5]. In addition, the engineering workstation serves as a simulation controller, allowing the user to select initial conditions and the type of aircraft to be simulated, and to set various parameters associated with the research being performed.

Projected Simulator Configuration

The adaptation of the Navy A-6E Operation Flight Trainer for flight dynamics instruction and research involves a series of modifications to the existing system. An incremental change methodology was adopted because it eliminated the long down-time associated with a complete rehost. It was understood that the simulator would serve an important public relations function, and therefore the capability was retained to easily return the simulator to its original configuration for demonstration purposes. The current architecture, in which the original Gould 32/67 is retained, reflects these decisions. The original warm-start OFT code, modified only to accommodate hardware changes (such as the control loader), may be used at any time to revert to the original capabilities.

Figure 2 shows the projected simulator configuration in the instructional and research mode. In the demonstration mode, the new computer subsystems are used solely to enhance flight visualization for onlookers.

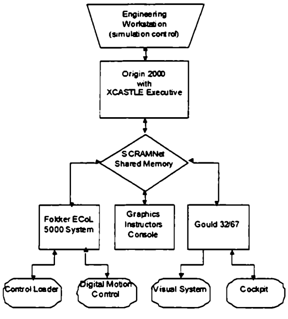


Figure 2 : Projected Simulator Configuration

From figure 2 one sees that the Origin 2000™ system is the master computer and that the Gould 32/67 has been demoted to an I/O slave. The Gould 32/67's primary tasks are communication with the cockpit, motion system, and the visual systems' Evans & Sutherland SP1 Image Generator. Communication between the Origin 2000™ and the Gould 32/67 is accomplished with the aforementioned SCRAMNet® shared memory system. Similarly, a graphics instructors console communicates with the Origin 2000™ utilizing SCRAMNet®. The Fokker ECoL 5000 system is shown controlling both the motion and the control loader. The touch screen engineering workstation located in the cockpit serves as a graphical interface with the Origin 2000™ and thus the main controller of the simulation. Among other advantages, the projected configuration will afford Virginia Tech the capability to simulate multiple aircraft, the ability to run more computationally intensive flight control systems, and the capacity to make real-time modifications to the control stick dynamics.

Instructional Use

The simulator serves as an instructional tool in all flight dynamics and control related classes. Introductory flight dynamics classes teach in the classroom the physics and analysis methods. Visualization of the various aircraft response modes were previously demonstrated by waving a plastic model around in front of the students. Students now see these responses in the simulator from the point-of-view of the aircrew, presented simultaneously with the results available to an engineer.

Senior/Masters level Aerospace and Ocean Engineering students at Virginia Tech have the option of taking a technical elective in Aircraft Automatic Flight Control. This course incorporates the analysis of an aircraft's dynamic responses as well as the design and implementation of linear and non-linear stability and control augmentation systems. Currently, student analysis of aircraft responses is performed on desktop simulations using a simplified A-6E simulation model. This analysis is compared with flying qualities requirements, which forms the criteria for design of the automatic control system. Designs are refined on the desktop simulation, then incorporated in the 2F122A for testing. The use of a relatively unsophisticated model for design, and subsequent "flight testing" on a high-fidelity simulation, introduces the realities of control system design using uncertain models.

Research Example: Control Allocation

Aerospace researchers at Virginia Tech have made important contributions to the understanding of the control allocation problem. The problem concerns aircraft that incorporate multiple, redundant control effectors. Typical control configurations (such as the ACTIVE) may have upwards of ten redundant effectors, and a number approaching twenty is easy to imagine. The "optimal" allocation (also blending, mixing, or scheduling) of these effectors with consideration of their physical rate and position limits can be a difficult and computationally intensive problem. Virtually all the potential problems associated with the implementation of control allocation algorithms in actual flight control systems relates to the demands likely to be placed on the system. It is extremely difficult to generate realistic aircraft maneuvers without a real-time man-in-the-loop simulation. A primary goal of the research to be performed using the 2F122A is to assess the computational requirements of the algorithms in highly demanding flight regimes that range from air combat to tight-tracking tasks to Pilot Induced Oscillations (PIO).

One of the features of the currently researched control allocation algorithms is their ability to manage rate as well as position limiting. A potentially serious drawback of this feature is the possibility of control "windup". This may occur when long periods of sustained high-rate demands are made of the control effectors. The phenomenon is manifested by the controls being driven to positions far from that of zero deflection. A primary research objective is to evaluate the control windup characteristics of the control allocation algorithms under representative and worst-case maneuvering flight conditions.

Modern fly-by-wire or control-by-wire flight control systems operate by interpreting the pilot's control stick inputs in some way determined by the desired flying qualities of the aircraft. Multiple, redundant controls are then allocated in order to satisfy the demands implied by the control law's interpretation of the pilot's inputs. In general, there is no correspondence between the limits on the pilot's stick (rate or position) and the corresponding limits on the combined control effectors. In particular, the pilot may be able to deflect the control stick faster than the control effectors are capable of following. This lack of stick-effector correspondence can cause the pilot's stick inputs to become out-of-phase with the aircraft response and result in PIO. However, the instantaneous maximum rate of the control allocation law is known, and may be used to restore a one-to-one correspondence between stick inputs and control surface deflections and possibly mitigate these PIO tendencies.

It is expected that the restoration of a direct correspondence between pilot input and control effector deflection through control allocation rate limiting will have adverse effects on the flying qualities of the aircraft. This research seeks to evaluate these effects through pilot in-the-loop real-time simulation utilizing the 2F122A. The Fokker ECoL 5000 electric control loader will be used to dynamically vary stick rate and position limits. A flying qualities assessment utilizing experienced test pilots performing representative tasks will be implemented in order to determine whether the flying qualities are degraded. The results of these evaluations will be analyzed to determine the viability of real-time implementation of optimal control allocation.

Conclusions

This paper has given a brief introduction into the tasks involved with adapting a Navy 2F122A A-6E OFT for flight dynamics research and instruction. It was shown that a group with no substantial background in flight simulation can complete such tasks. Furthermore, this paper has presented some of the possible instructional and research capabilities that a high-fidelity flight simulator offers on the university level.

Acknowledgements

This project was sponsored by the Naval Air Systems Command, and by the Simulation and Control Technology Department of the Naval Air Warfare Center Aircraft Division, Patuxent River, Maryland.

AIAA-98-4181

A PORTABILITY STUDY ON IMPLEMENTATION TECHNOLOGIES BY COMPARING A RAILWAY SIMULATOR AND AN AIRCRAFT FTD

Sugjoon Yoon*
Sejong University, 98 Goonza-dong, Gwangzin-Gu, Seoul, Korea

ABSTRACT

The paper introduces major technical specifications of the Line II railway simulators of Pusan City in Korea. Comparing design specifications of the railway simulators with those of the light aircraft Flight Training Device (FTD), the paper reveals commonality of implementation technologies applied to both simulators: Overall configurations and design philosophies are basically the same. In both programs VMEbus computing systems with UNIX are adopted as backbones of the simulators. It is found that the railway simulators are less stringent in real-time requirements than the aircraft FTD and the railway simulators are designed to be more event-driven and object-oriented. The experiences show that models may be diverse depending on the objects but implementation technologies are about the same. Maximizing portability of implementation technologies is a matter of an organization's strategy of adopting standardized processes and modular technologies available and most economic to them.

Introduction

Flight Simulator Industry has lead overall simulation technologies since the appearance of Link's first Trainer. Most people in simulation communities today understand or at least have some knowledge of, general requirements and technical specifications of airplane simulators. On the other hand, railway simulators have begun to attract customers' interests and formed a niche market since the beginning of 1980's. Information regarding railway simulators such as requirements, technical specifications, modeling, system components, etc is relatively unknown to simulation communities, and belongs to manufacturers of railway simulators or special research institutes. This paper reveals technical specifications of railway simulators in relative detail, which are being tested for practical use in Korea, and studies portability of simulation technologies by comparing the technical specifications with those of flight simulators. For practical comparison, technical specifications of CG-91 Flight Training Device[1,2,3] and PUTA (Pusan Urban Train Association) Line II Metro Simulators[4] are used as examples. Both of the simulators were developed by Korean Air between 1991 and 1998.

Railway Simulators

PUTA Line II Metro Simulators provide trainees with realistic driving environment of the Pusan City metro system and enable effective and safe training. The railway simulators are composed of a pair of cabs, which means two metro drivers, one at each cab, can be trained independently at the same time by an instructor at a common control station.

Training through railway simulators is categorised into basic and application courses, which is a little more elaborated in Table 1. The basic training courses encompass manipulation of on-board equipment and familiarisation of track environment including signals on route. Their goals are familiarisation with operation environment. The application courses include abnormal situations in addition to the basic training so that trainees can be trained and tested for their responses. Relevant training results and their evaluations are provided to trainees and an instructor either in printed report forms or on screen. A CCTV camera is installed in each cab and records whole training processes for later use during performance evaluation of a trainee.

Interior and exterior of PUTA Line II Metro Simulators are shown in Figs. 1 and 2. H/W and S/W configurations of a set of railway simulators are illustrated in Figs. 3 and 4. As shown in the figures, the system hardware is composed of a pair of cabs and relevant electric signal and power lines, which are almost symmetric and connected to an Instructor Operation Station (IOS).

* Associate Professor, Vice President, Aerospace Industry Research Institute, School of Mechanical and Aerospace Engineering. Member AIAA.

Table1 Training courses of a railway simulator

Basic Training	Application Training
Initial starting procedures	Basic Training
Start, operation, stop	Emergency procedures
Manipulation of on-board equipment	Malfunctions
Response to signals on route	Operation and stop on emergency conditions
Route familiarisation by visual display of forward track views	

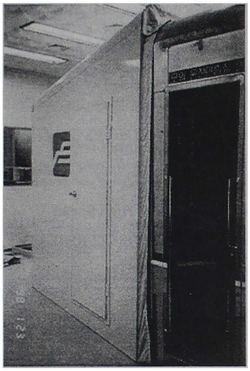


Fig. 1 Exterior of PUTA Line II metro simulators

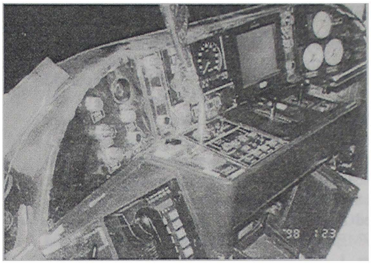


Fig. 2 Interior of PUTA Line II metro simulators

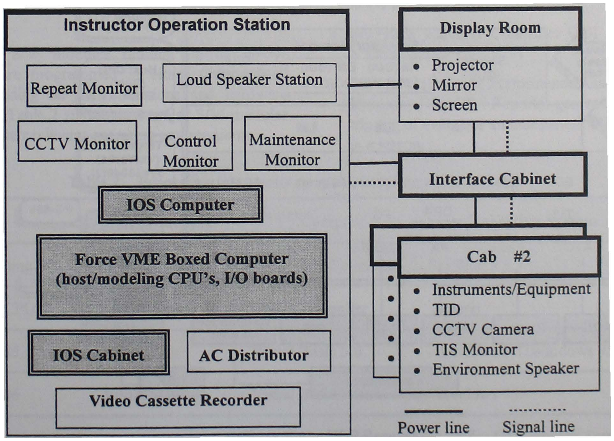


Fig. 3 H/W configuration of PUTA Line II metro simulators

The followings summarize configurations of simulator components and their functions shown in Fig. 3.

■ IOS computer

An IOS computer is used for a common terminal to the host computer and the operation environment of Graphical User Interface (GUI) group S/W. The monitor of the IOS computer is used as a control monitor to show IOS menus, and as a terminal of the host computer for maintenance purpose. Its technical specifications are as follows:

- Model: COMPAQ DESK PRO 2000
- Operating System: Window 95 or Window NT
- Comm. Protocol : TCP/IP

■ Host CPU

A host CPU or a host computer is for system maintenance and operation environment of S/W modules belonging to the host group in Fig. 4. Each of execution files and data files is controlled and monitored by the host CPU, which is a part of a Force VMEbus computer system. The system comprises 1 host CPU board, 2 modelling CPU boards, and several I/O boards. The boards in the VME rack communicate with each other on VMEbus. Relevant system specifications are as follows:

- Model: Force VME Boxed System
- CPU: Sun SPARC 5V
- Operating System: Solaris 2.1(SunOS v4.1)
- Comm. Protocol : VMEbus, TCP/IP

■ Modeling CPU's

2 Modeling CPU's are for running S/W modules which belong to the modeling group in Fig. 4. Each of the CPU's independently drives relevant one of 2 cabs. Their connections to the cabs are almost symmetric, and malfunctions of a CPU affect the other CPU's or cab's operation to a minimum. That is, training of two cabs is coupled minimally. Major specifications are

- Model: SYS 68K/CPU-40B
- CPU: MC68040, 25 MHz
- DMA Controller: 32 Bit High Speed
- Serial Interface: RS232/RS422/RS485
- VMEbus Interrupter: IR 1-7

■ Visual Computers

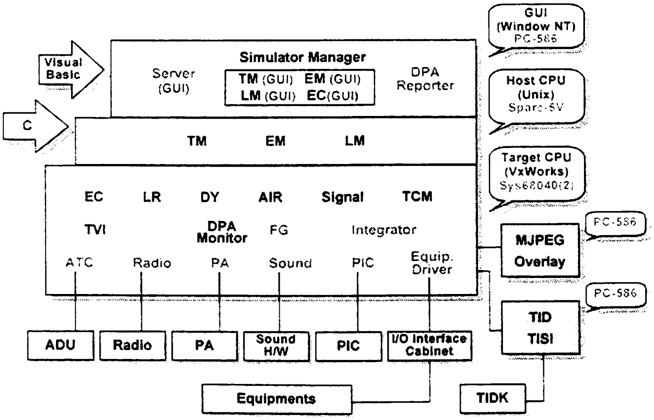


Fig. 4 Overall S/W configuration

A pair of visual computers are located in the IOS cabinet. A visual computer for a cab receives position data of a train via Ethernet from the TVI(Track View Information) module on the relevant modeling CPU, and retrieves a visual image, front track-view, which is the most appropriate for the position among the filmed visual database compressed on computer hard disks. Motion JPEG is used for the purpose. A track-view image in the form of Motion JPEG is transmitted to an overlay card, and superimposed by light signals generated with computer graphics. The resultant visual image is finally transmitted to the repeat monitor in the IOS and the visual projector for a trainee in a cab. Major specifications are as follows:

- Model: COMPAQ DESK PRO 2000
- Overlay Card: Digi-Mix
- M-JPEG board: Digi-Motion

  (9 GB hard disks included)
- Operating System: Windows 95 or Windows NT

■ TIDK-TID driving computer

A pair of TIDK-TID driving computers is also located in the IOS cabinet as the visual computers. One computer per cab is for generating signals to TID (Track Information Display) and TIDK (Train Information Display and Keyboard). The specifications are

- Model: COMPAQ DESK PRO 2000
- FIP control box

All the S/W modules running on the above computers are programmed in either C or Visual Basic depending on their functions and hardware platforms. Table 2 summarizes major S/W modules along with their relevant operating environments.

■ Database

Major data files used by execution programs of PUTA Line II simulators are categorised into

- track data
- performance data of the trains
- electric circuit data of relevant on-board equipments
- pneumatic data of relevant on-board equipments

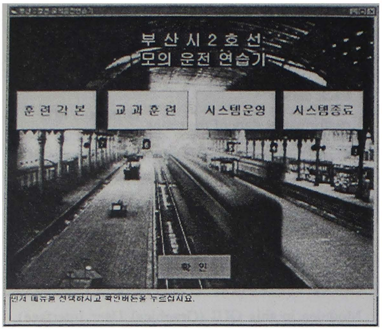
■ GUI group (IOS) S/W

Modules in this GUI (Graphical User Interface) S/W group are programmed in Visual Basic 5.0 of Microsoft in order to decrease development process as well as its time and increase maintainability of the modules. With on-screen menus representing the GUI group S/W an instructor controls and monitors the whole system including 2 cabs. The GUI group S/W is a window to relevant S/W modules in host and target CPU's. An example of on-screen menus is shown in Fig. 5 and a part of the IOS menu hierarchy in Fig. 6. The GUI group S/W is composed of following sub-components:

- TM(GUI): Track Manager GUI invokes TM in the host CPU.
- EM(GUI): Exercise Manager GUI invokes EM in the host CPU.
- LM(GUI): Lesson Manager GUI invokes LM in the host CPU.
- EC(GUI): Exercise Controller GUI invokes EC in the target CPU.
- DPA RPT (Driver's Performance Assessment Reporter): The DPA Reporter generates either a trip-log of events or an assessment of the events in a lesson.
- Server(GUI): Server GUI invokes Server.

Table 2 Programming languages and operating environments

	S/W module	Compiler	Operating System
Modeling CPU	Master Driver #1	GNU .68KC compiler	VxWorks 5.3
Host CPU	Master Driver #2	GNU C compiler	Solaris
PC-586	IOS S/W	Visual Basic 5.0	Windows NT/Windows 95
PC-586	Visual S/W	Visual Basic 5.0	Windows 95



먼저 메뉴를 선택하시고 확인버튼을 누르십시요.

Fig. 5 An example of IOS menu on screen

■ Host group S/W

The host group S/W in Fig. 4 is composed of TM (Track Manager), EM (Exercise Manager), and LM (Lesson Manager). These components are common to a pair of simulator cabs and do not communicate frequently with modeling group S/W once training lessons begin. They take important roles as lessons begin or are completed. The S/W components and their functions are listed below:

- TM (Track Manager): TM is comprised of 3 interlocking tasks, manage track tections, manage media and manage routing, which combine to handle the building and amendment of track sections, media information, and route validation and selection.
- EM (Exercise Manager): EM manages lessons and comprises Exercise Builder and Exercise Management Facilities. Exercise Builder generates a table with the route information TM provides, in order for the other modules to refer. Exercise Management Facilities changes exercises depending on selected conditions such as faults and signals.
- LM (Lesson Manager): LM supplies exercise signal information during initialization of lesson. Exercise signals consist of id, distance, type, state, fog repeater indicator, auto signal indicator and distance/speed/state that the auto signal is checked against.

■ Modeling group S/W

Pairs of S/W modules in the Modeling group operate independently on two symmetrical modeling CPU boards. S/W modules belonging to this group are as follows:

- EC(Exercise Controller): EC synchronises tasks in a lesson and logs events generated during the lesson and controls a snapshot.
- LR(Lesson Reviewer) : LR reviews lessons performed.
- DY (Dynamics): DY supplies train position and speed. The DY model calculations can be split into four main areas such as longitudinal, carriage weight, cab distance, and train motion.
- AIR (Air System): The air system model provides six basic functions such as initialization,

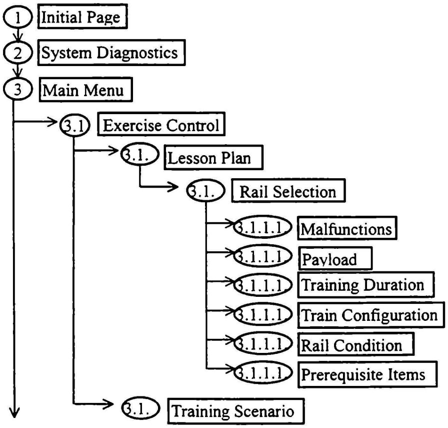


Fig. 6 An example of IOS menu hierarchy

Table 3 A list of on-board equipment

No	Switches/Indicator/Equipment	Interface
1	Master Controller	AI
2	TRCP(Train Radio Control Panel)	RS485
3	window wiper switch	DI
4	MDH(Mode Direction Handle)	DI
5	ADU (Automatic Display Unit)	AO
6	dimmer switch	AI
7	push button switch (3EA)	DI
8	push button switch (11EA)	DI
9	indicating lamp (4EA)	DO
10	duplex pressure gauge	AO
11	DC volt meter (3KV)	AO
12	DC volt meter (150KV)	AO
13	PIC (Passenger Information Controller)	RS485
14	TIS (Train Information System)	RS485
15	NFB	DI & DO
16	head lamp control switch	DI
17	whistle valve with foot pedal	DI
18	lamp push button switch	DI
19	destination indicator	DI
20	train number indicator	DI
21	emergency brake cut out switch	DI
22	rescue operating switch	DI
23	emergency brake switch	DI

control, air pressure model, response to opening and closing valves, generation of outputs on pressure thresholds, and calculation of air braking force in response to requests of dynamics.

- SIG (Signal): All signals are initialized to a default state of having the topmost bulb set. If this is not the required state then it is changed from the graphical user interface.
- DPA Mnt (Driver's Performance Assessment Monitor): DPA Mnt provides events that are used when assessing the drivers performance. The DPA monitor is provided with the speed limits, stop zones, their positions, and obstacles on the track.
- TVI (Track View Interface): TVI provides an interface on Ethernet between IOS and visual computers. It converts data/control signals of Master Driver into the format required by the visual computers.
- TCM (Train Circuit Modeler): TCM models the circuit diagrams in such areas as door control, round train & safety, traction & braking, driving & auxiliary control, train lines & car wires, and a compressor system.
- FG(Fault Generator): FG generates faults or malfunctions of on-board systems or signals, which can be selected or deselected either interactively or at a specific distance selected during definition of an exercise.
- Integ (Integrity test): Integ diagnoses system problems.
- ATC: ATC(Automatic Train Controller) emulator
- Radio: radio driver
- PA: PA(Passenger Address) driver
- Sound: This simulates sounds normally generated by equipment in the cab or the environment. The audio system is controlled by the Exercise Controller.
- PIC (Passenger Information Controller): PIC driver
- Equip Drv.: On-board equipment drivers

■ Cab #1, #2

In simulation of switches, indicators, and equipment in the cabs an applied principle is to minimise their modification. If major modification is necessary, the actual system is replaced by a digitally controllable simulated system. For some equipment such as Passenger Information Controller and TIDK (Train Information Display and Keyboard) stimulation is rather adopted. That is, signal inputs to the equipment are simulated. Table 3 is a list of switches, indicators, and equipment simulated or stimulated, where AI (analog input), AO (analog output), DI (digital input), and DO (digital output) mean interfacing ports of VME I/O boards which drive the switches, indicators and instrument.

Airplane Simulators

In order to study portability of S/W and H/W in simulators technical specifications of PUTA Line II metro simulators are introduced above. For comparison technical sepcifications of airplane simulators are briefly described here. The airplance simulator selected as an example is a flight training device developed by Korean Air between 1991 qand 1995 to train pilots of a light aircraft, ChangGong-91. The aircraft was also developed by Korean Air in 1991. The flight training device is shown in Fig. 7. Its S/W and H/W configurations are illustrated in Figs. 8 and 9.

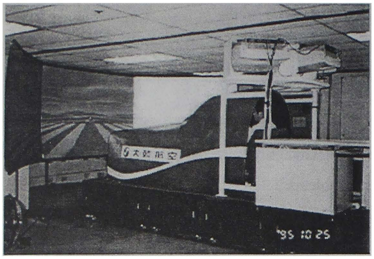


Fig. 7 ChangGong-91 Flight Training Device

Portability study of S/W and H/W

In this paper parts of technical specifications of a railway simulator and a flight training device are briefly introduced. The portability of H/W and S/W concentrated in the paper is about computer H/W, S/W, and IOS, which are actually major components of simulators. They really determine performance and value of a simulator.

Let us investigate computer H/W of PUTA metro simulators. The host and modeling computers in the simulator are of VMEbus type. VMEbus computers are also used for the ChangGong-91 FTD, and frequently chosen for computer systems of airplane, ship, and automotive simulators. Complexity of VMEbus cards' combination depends on application. As higher fidelity simulation is required, or as the simulated system is more complicated, higher performance CPU boards and more I/O channels may be included in the VMEbus computer. That is, complexity and fidelity of simulation determine configuration of a VMEbus computer. That is why different configurations of VMEbus computers are applied to PUTA metro simulators and ChangGong-91 FTD. However, computer-related technologies during implementation such as HILS (Hardware-In-the-Loop-Simulation) technology and relevant equipment, are basically the same either in railway or airplane simulators.

S/W such as real-time OS, I/O drivers, and communication S/W is closely related with computer H/W, and its portability depends on the computer H/W. Such S/W may be amended partially in order to meet interface requirements of modeling S/W and simulated/stimulated equipment. However, accumulated technologies through experience of system developments allow easy alternation of the S/W. Let us compare Figs. 3, 4, 8, and 9. Similarity of H/W and S/W configurations of the railway simulator and the FTD is revealed. S/W of an airplane simulator is stricter to real-time requirements and rapider response time is expected. In case of a railway simulator these constraints are less stringent. That is a rational of applying an executive-type time controller to an airplane

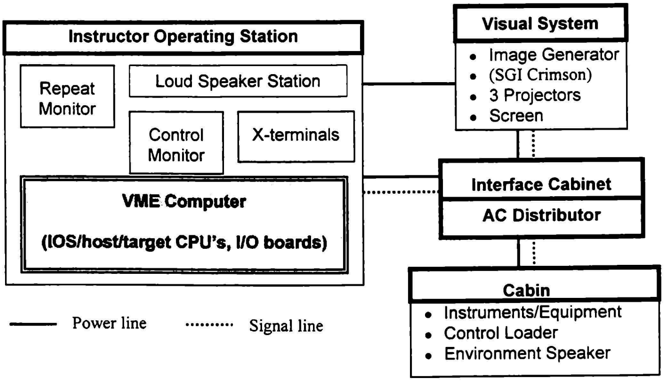


Fig. 8 H/W configuration of ChangGong-91 FTD

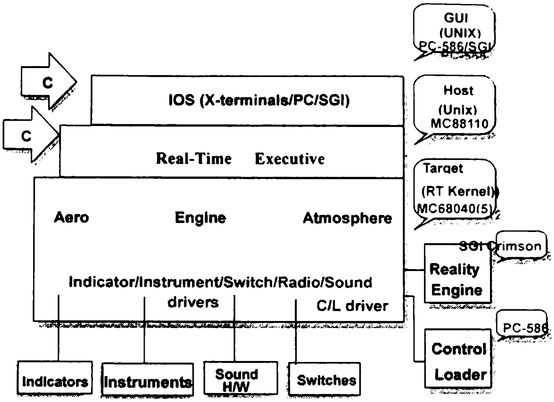


Fig. 9 S/W configuration of ChangGong-91 FTD

simulator and an event-driven scheduler to a railway simulator. However, it is considered reasonable to adopt executive-type scheduler even in a railway simulator if simulation model is stricter and more sophisticated, and if exact response characteristics is required.

In case of IOS menu designs similar observation is obtained. Menu contents can be very diverse depending on simulated systems and customer's requirements. However, S/W tools, programming languages, and menu hierarchies to realize IOS menus are about the same. Standardized procedures

		Flight Simulator		Railway Simulator	
		Typical	CG-91	Typical	PUTA II
H/W	Host Computer	W/S VMEbus (off-the-shelf set)	VMEbus (assembled)	VMEbus (off-the-shelf set)	VMEbus (off-the-shelf set)
	Visual system Image generator Display System Database	Exclusive CGI Collimated CG	Graphic W/S Non-collimated CG	LD⇒ CGI Non-Collimated Video film⇒CG	CGI+MJPEG Non-Collimated Video film, CG
	Cab	Simulated	Simulated	Simulated	Actual
	Motion Platform	6 DOF	fixed	fixed, 4 DOF	fixed
	IOS	Graphic W/S	X-term, PC, W/S	PC	PC
S/W	Language	Fortran, C	C	C, C++	Visual C, Basic
	OS	Exclus.⇒Unix POSIX comp. Executive type	VMEexec Non-POSIX Executive type	Unix derivatives POSIX comp. Event-driven	VxWorks POSIX comp. Event-driven
	Comm. Protocol	Ethernet, bus	Ethernet, bus	Ethernet, FTP	Ethernet, FTP
	System Model	Accurate Test data	Accurate Test data	Simplified Actual+Generic	Simplified Actual+Generic
	IOS Menu	GUI	GUI	GUI	GUI
Relevant Regulations		FAA, CAA, etc.	FAA, CAA, etc.	none	none

Table 3 Comparison of H/W and S/W configurations of railway and airplane simulators

can be set and applied independent of simulated systems based on implementation experiences. Apparent features and operating procedures of IOS' may look quite different from one another. But striking similarities in training procedures and IOS-inherent features, which enable effective training, are observed between railway and airplane simulators. For example, configuration, concept and methodology behind training lessons, normal and emergency procedures are very close to each other.

Conclusions

The experiences show that models may be diverse depending on the objects but implementation technologies are about the same. Maximizing portability of implementation technologies is a matter of an organization's strategy of adopting standardized processes and modular technologies available and most economic to them. Table 4, which compares technical specifications of railway and airplane simulators, makes this conclusion more obvious.

References

[1] Yoon S. et al, Final Report for Development of a Flight Training Device, Korean Ministry of Commerce and Industry, June 1995.

[2]Yoon S., Kim W., and Lee J., "Flight Simulation Efforts in ChangGong-91 Flight Training Device", Proceedings of AIAA Flight Simulation Conference, Aug. 1995.

[3] Yoon S., Bai M., Choi C., and Kim S., "Design of Real-Time Scheduler for ChangGong-91 FTD", Proceedings of 'Making it Real' CEAS Symposium on Simulation technologies, Delft Univ., Nov. 1995.

[4] PUTA II Simulator Requirements Specification, Doc #. SIM-001-002, Korean Air, May 1997.

AIAA-98-4182

THE MULTI-CHANNEL CIG SYSTEM OF A MANNED SPACECRAFT SIMULATOR

Wang Yuefeng*, He Jiamin†, Huang Keli‡, Ren Xuan §
Department of Automatic Control
National University of Defense Technology
ChangSha , Hunan 410073
P.R.China

This paper presents the low-cost CIG system of a manned spacecraft simulator, addresses some of the issues in real-time computer image generation. It establishes the coordinate transformations among the different coordinate systems. The areas on the celestial sphere that can be seen from a spacecraft are also discussed. To develop the CIG system, 3D computer graphics, rendering algorithms, parallel processing and network communication are being taken. To provide greater realism, GL, a 3D toolkit made by SGI is used. The characteristic is to use commercially available general purpose personal computer and graphics workstations instead of special purpose hardware and image generator combination.

1.Introduction

The CIG system includes five computer, the first is used for instrument panel simulation, the second and third are used for side-windows scene, the fourth is used for periscope scene, the 5th is a network management computer. Communication among computers is accomplished by using the Berkeley socket mechanism. This CIG system uses cheap computer but not high performance hardware graphics engines. The key thing is that there are significant technological limitations to the underlying hardware. It is the design and construction of the software that can compensate for these limitations. The user interacts with a synthetic world through a computer-image-generated scene. The challenge is how good of a scene can be presented and how fast. In order to keep the amount of geometry being sent through the graphics pipeline at an acceptable level, the graphics primitives must be developed and stored in a format that allows the software to render all of the geometry that is in the view frustum.

2.Scene Management

There are three scenes the astronaut can see through the side-windows or the periscope.

. the sky and some stars
. part of the Earth
. part of the Earth, sky and some stars.

A star chart database and a earth database are constructed, the primary function of scene management is the selection of the stars to be passed to the rendering process. The process of selecting the geometry to be displayed is called culling. It should take less time to cull geometry out of the database than it does to draw them. for examples, if the culling takes place at the primitive level in a large simulation database. The system could fail to maintain the desired frame rate. This is due to the computational time required to check each primitive in the entire database to see if it should be rendered in the current scene. On the other hand, if no culling take place, then every primitive in the database must draw for every frame and the graphics pipeline will become the limiting factor.

2.1 Coordinate Systems

(1) geocentric-equatorial coordinate system O_E-$X_EY_EZ_E$

The geocentric-equatorial coordinate system is an inertial coordinate system, the origin of the coordinate system is at the center of the Earth, the X_E axis lies along the intersection of the Earth's equator and the plane of the Earth's orbit around the Sun(ecliptic), This intersection point is called the vernal equinox, the Y_E axis passing through 90 degrees East longitude at the Equator, and the Z_E axis passing through the North celestial pole, as shown in Figure 1.

(2) Local geographic coordinate system O_l-$X_lY_lZ_l$

The geographic coordinate system usually has its origin at the center of spacecraft mass, $\bar{r}$ is distance vector from O_E to the spacecraft, the X_l axis pointing in the direction of vector $\bar{r}$, the Y_l axis pointing east and the Z_l axis pointing north, The relationship with geocentric-equatorial coordinate system are illustrated in Figure 2. The α and δ in Figure 2 is the right ascension and declination of spacecraft.

(3) Orbital coordinate system O_o-$X_oY_oZ_o$

The orbital coordinate system has its origin at the center of spacecraft mass, the X_o axis points in the direction of the movement, the Y_o axis points in the direction of vector $\bar{r}$ and the Z_o axis is orthogonal to the other two axes in a right-handed system, as shown

* Associate Professor
† Assistant Professor
‡ Professor
§ Professor, senior member

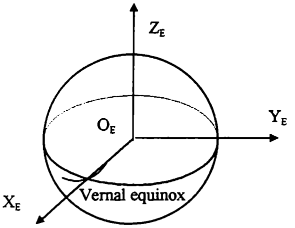


Figure 1. Geocentric-equatorial coordinate system

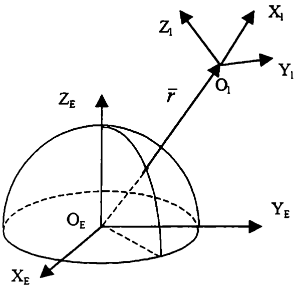


Figure 2 . Local geographic Coordinate System

in Figure 3.The angle A between X_o and Z_1 is called azimuth angle.

(4) Body coordinate system O_b-$X_bY_bZ_b$

The body coordinate system is a Cartesians coordinate system fixed to the spacecraft, it has its origin at the center of mass, the positive direction of each axis extends out the front(X_b axis),the up(Y_b axis),and right(Z_b axis)

According to the definitions of above coordinate systems, we can get rotation matrix(direction-consine matrix) between them.

$$M_{E-1}=\begin{pmatrix}\cos\delta & 0 & \sin\delta\\ 0 & 1 & 0\\ -\sin\delta & 0 & \cos\delta\end{pmatrix}\begin{pmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}X_1\\ Y_1\\ Z_1\end{pmatrix}=M_{E-1}\begin{pmatrix}X_E\\ Y_E\\ Z_E\end{pmatrix}\qquad(1)$$

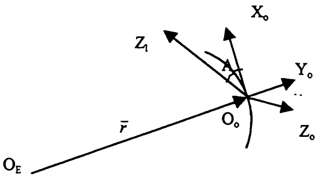


Figure 3. Orbital coordinate system

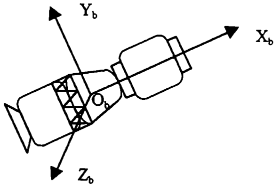


Figure 4.Body coordinate system

$$M_{1-0}=\begin{pmatrix}-\sin A & 0 & \cos A\\ 0 & 1 & 0\\ -\cos A & 0 & -\sin A\end{pmatrix}\begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}X_0\\ Y_0\\ Z_0\end{pmatrix}=M_{1-0}\begin{pmatrix}X_1\\ Y_1\\ Z_1\end{pmatrix}\qquad(2)$$

$$M_{o-b}=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\varphi & \sin\varphi\\ 0 & -\sin\varphi & \cos\varphi\end{pmatrix}\begin{pmatrix}\cos\psi & 0 & -\sin\psi\\ 0 & 1 & 0\\ \sin\psi & 0 & \cos\psi\end{pmatrix}*\begin{pmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}X_b\\ Y_b\\ Z_b\end{pmatrix}=M_{o-b}\begin{pmatrix}X_0\\ Y_0\\ Z_0\end{pmatrix}\qquad(3)$$

Where θ,ψ,φ are the pitch ,yaw and roll angles ,being illustrated in Figure 5 .

2.2 Field of View Determination

When a human moves around, the portion of the world that he see changes. The view volume is a

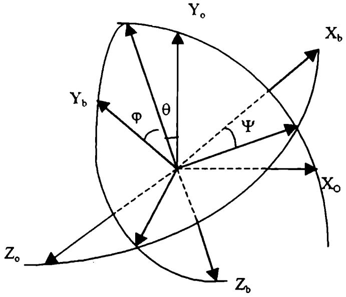


Figure 5. Pitch, yaw and roll angles

function of the human's position, orientation of the head, atmospheric conditions, and time of day levels. In spacecraft simulator, the same parameters should be taken in consideration during the construction of the scene. According to the position and attitude of spacecraft, We can calculate the visible stars on the celestial sphere and continent on the Earth by coordinate transformations. Here takes the side-windows scene as an example, the periscope scene runs in the same way.

In the body coordinate system, the right side-windows is in the direction of Z axis, and the left side-windows is in the negative direction of Z axis. Assuming the astronaut's field of view angle is 2*ν ,ν<90 degrees. Using the geometric relationship shown in Figure 6,we can solve for μ1 and μ2 as:

$$\mu 2=\arcsin(Re/r) \qquad (4)$$

where:

Re radius of the Earth

r the distance from the center of the Earth to spacecraft

$$\mu 1=\mu-\mu 2 \qquad (5)$$

when μ1>ν just stars can be seen

μ1<-ν just the Earth can be seen

-ν<μ1<ν stars and the Earth can be seen

The angle μ can be determined .In the body coordinate system, the Z_b vector {0 0 1 } (corresponding to the right side-window scene) transform to orbital coordinate system use:

$$\begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} = M_{o-b}^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad (6)$$

The direction consine of the vector with Yo axis use:

$$\mu=\pi-\arccos(Yo) \qquad (7)$$

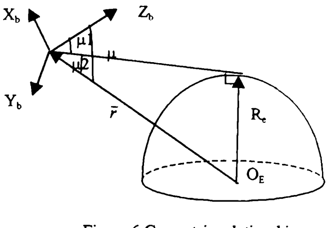


Figure 6.Geometric relationship

When we know some stars can be seen from side-window as discussed above, a brute force method is to draw every star in the stars database. Clearly, the brute force approach is not the most efficient way. To determine visible areas on the celestial sphere, we first compute the intersection point of Z_b axis with the celestial sphere. Because it is very smaller that the distance from the center of the Earth to the spacecraft compare with the distance from the star to the spacecraft, the view error is not being take considered, we simply put the spacecraft in the center of the celestial sphere, the Z_b is converted to geocentric-equatorial coordinate system through a series of transformations. from (1)、(2)、(3), We have

$$\begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = M_{0-b} \bullet M_{1-0} \bullet M_{E-1} \begin{pmatrix} X_E \\ Y_E \\ Z_E \end{pmatrix} \qquad (8)$$

the right ascension and declination are computed using:

$$\begin{cases} \alpha_b = arctg(Y_E / X_E) \\ \delta_b = \arcsin(Z_E) \end{cases} \qquad (9)$$

Once the right ascension and the declination of the intersection point have been computed, they are used to calculate the visible areas, From Figure 7,we have

$$\Delta\delta=\nu \qquad (10)$$

$$\Delta\alpha=\arcsin(\sin\nu/\cos\delta_b) . \qquad (11)$$

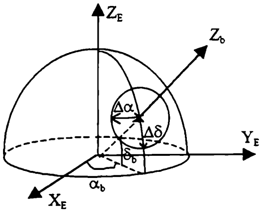


Figure 7. Visible areas on celestial sphere

2.3 Stars database

The precise format of an efficiently structured database is dependant upon the culling method used by the system. The stars must be modeled in a way to keep both the culling time and the flow of data in graphics pipeline at acceptable levels, The requirements of index efficiently and storage efficiently go together for any database. Unfortunately, they often are at cross with each other. The classic data structures example of this is the linked list versus the array. The linked list is flexible and expandable, but requires O(n) access time. The array, on the other hand, is a fixed construct in terms of size and structure, but can be accessed in O(1) time. For the distribution of stars on the celestial sphere is not even, both linked list and array are used. To segment the celestial sphere, first from declination -89 degree to 89 degree spacing 1 degrees, this results 178 rings. Second segment 180 blocks on every ring ,the post spacing is 2 degrees, as shown in Figure 8.The north celestial pole and south celestial pole are both a block(1 degree radius).The stars data structures are below:

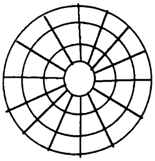


Figure 8.Segmantation of a half celestial sphere

```
struct star_struct  {
    float star_longi;   // right ascension
    float star_lati;    // declination
    float star_grade;
    struct star_struct *next_star;
 };
 struct star_array_head {
    int star_number;
    struct star_struct *next_star;
 };
 struct star_array_head
      star_mesh[178][180],
      star_south_pole,star_north_pole;
```

3. 3D Graphics rendering

Rendering,or drawing, involves the transformation of the graphics primitives selected by the scene management process into pixels on The screen. The visual realism of the scene and the frame rate are the two most important parameters in the display .

In summary, the graphics programs are written in C call the Silicon Graphics' Graphics Library(GL).The GL is a set of graphics and utility routines that provide high- and low-level support for graphics. It includes depth-cued, antialasing, belending, lit, shading, texture mapping, atmospheric effects and so on.

There are many ways to do hidden surface removal. The method used in GL is a Z-buffer. Operations required to do hidden surface removal at each pixel would dramatically reduce performance. When rendering a object, using visible surface detecting method will reduce rendering costs. Here taking sphere as a example.

Usually, sphere surface is divided to many planes of quadrangle or triangle, drawing these planes ordinal ,a sphere is completed. If GL lighting model such as surface normal, material properties, and light sources are defined, a realistic sphere is drawn. Further more, if GL texture mapping features are used ,a photo-realistic earth is rendered. But in fact, about a half of polygons of a sphere is not visible. If the hidden polygons don't be drawn by means of visible surface detecting, instead of Z-buffer , the frame rates will increase. To determine a polygon that consists a sphere is visible or not,the angle between the polygon's normal line and line of view must be computed. If the angle is greater than 90 degrees, the polygon is not visible, it is not necessary to draw this polygon. As the Earth rotates(in GL, called modeling transformations),the spacecraft moves, and the view point and line of view change(in GL, called viewing transformations). As shown in Figure 9.the angle can be calculated

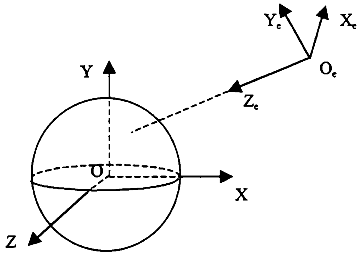


Figure 9. Modeling and viewing transformations

By analysing the GL drawing process,the coordinate transform calculation can be reduced. The eye coordinate system is the result of object coordinate system transformed by the modeling and viewing matrix, as shown in Figure 10,GL concatenates modeling and viewing transformations on a single matrix(ModelView).We can utilize the ModelView matrix for hidden polygons removal.

In fact, it is not necessary to calculate the value of the angle between the normal line of a polygon and the line of view. That computing the normal line's Z value in the eye coordinate system is enough. If the Z is positive, this polygon is visible, and if negative, this

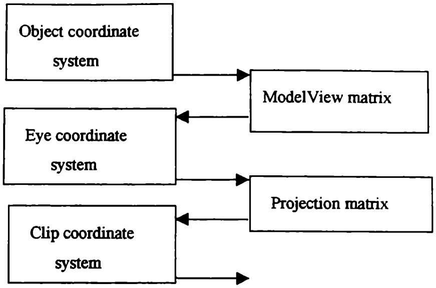


Figure 10 . GL Coordinate System

polygon is not visible. As shown in the code fragments below:

```
           ⋮
    mmode(MVIEWING);
           ⋮
    Matrix m;
    getmatrix(m);
           ⋮
    if(surf_visible(surf[i][j],m)) {
      //  the polygon is visible, drawing
    }
    else {
      //  the polygon is not visible,
      //  ending quadrilateral strips
     }
          .⋮
 int surf_visible(float normal[3],Matrix m)
 {     float z;

z=normal[0]*m[0][2]+normal[1]*m[1][2]+normal[2]
*m[2][2];
   if(z<0)   return 0;
   else       return 1;
}
```

The GL quadrilateral strips routine is used for rendering. The advantage of this method is the fast rendering. Each grid post is passed to the rendering pipeline twice, rather than four times. The Sun provides a light source for realistic shading of the Earth.

4. Instrument panel

The simulant instrument panel runs on a micro-computer which has a graphics accelerator , a big screen monitor and a touch screen. There are navigational chart, warning light, digital dial, control buttons, clock and other in-vehicle devices on the screen. The driver can steers the spacecraft by interacting with the touch screen. The graphics accelerator is the IRISVISION board set from Silicon Graphics Inc. IRISVISION products utilize the GL software interface, which allows for the creation of high performance visual applications. The hardware features include Z-buffer ,double buffer, gouraud shading and so on. Using the IRIS GL and hardware features we can easily realize the function of instrument panel. For example, We draw static objects such as dial on underlay planes and movable objects such as pointer on normal planes.

5. Network management

The network is the ideal mechanism to increase the sacleablity and adaptability of the system. The network management computer (NMC) associates with above image generation computers in turn. In Figure 11,a data flow (solid line)diagram shows some of the state and control data that have to be passed from computer to computer. The NMC reads spacecraft's state data, which includes position, velocity, and attitude information from a spacecraft dynamics simulation computer and sends the data to the image generation computers. When the driver steers the spacecraft by touching the control button on the screen of instrument panel computer, the message data will be sent to the dynamics simulation computer through NMC. The spacecraft's state will vary with the control data. To avoid network data losing, data transfer is accomplished by means of handshake. When the current frame is complete on one image generation computer, this computer will apply for next frame data by sending a flag to NMC (dotted line). The NMC poll each image generation computer sequentially, when it gets a requisition, a state data is returned. The network uses the Ethernet network and the TCP/IP protocol. The NMC runs on a personal computer .It also store data in a file and can play back later for post mission analysis.

6.Conclusions

This CIG system provides multi-channel scenes on different computers, so it is flexible. It takes the advantage of parallel processing to reduce rendering

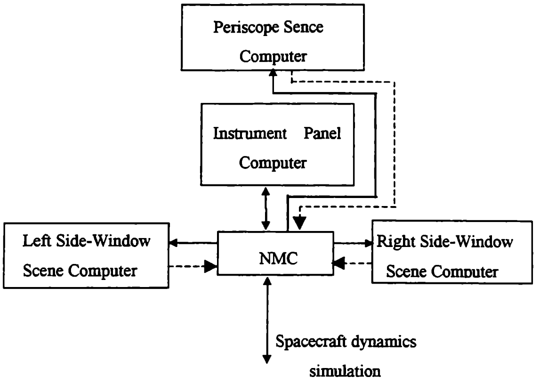

Periscope Sence Computer
Instrument Panel Computer
Left Side-Window Scene Computer
NMC
Right Side-Window Scene Comnuter
Spacecraft dynamics simulation

Figure 11 .Structure of the CIG system

Load. The CIG system is developed on PC and workstation. By using commercially available computers it is possible to reduce the cost of hardware development over many users. By using a computer as the hardware base, we further reduce the cost. Since the organization can use the computer for other purpose when it is not involved in a mission exercise. This is something that the special purpose hardware can not do. If user cannot afford to spend money to provide each simulator with high performance graphics workstation, this solution allows a compromise between reality and resources. In some case ,graphic rendering algorithms can be used for accelerate the output .

Acknowledgment

A large number of people have contributed to this work, in particular, special thanks go to Chenlei, Duohong, Liulian, and Caillin.

References

1. Ren Xuan.(1988) Satellite Orbital Dynamics. NUDT, ChangSha.

2. Silicon Graphics, Inc.(1990) Graphics Library Programming Guide. Mountain View, California.

3. David R.Pratt. A Software Architecture for Construction and Management of Real-Time Virtual Worlds. Naval postgraduate school. June 1993.

4. Roy David Young. NPSNET-IV: A Real-Time,3-D Distributed Interactive Virtual World. Naval Postgraduate School. September 1993.

5. Gary R.Smith, Marvin LeBlanc. The Distributed Earth Model Orbiter Simulation(DEMOS) Three Dimensional System at Johnson Space Center. SIMULATION, July 1993.

6. Martin R.Stytz and Andrea A.Kunz. Distributed Virtual Environment for Satellite Orbital Modeling and Near-Earth Space Environment Simulation and Portrayal. SIMULATION ,July 1996.

AIAA 98-4576

THE DRAFT AIAA FLIGHT MECHANICS MODELING STANDARD: AN OPPORTUNITY FOR INDUSTRY FEEDBACK

Bruce L. Hildreth
Vice President
Science Applications International Corporation
California, MD

Abstract

The Modeling and Simulation Technical Committee is a nationwide committee of approximately 30 members. It is a permanent committee whose charter is to promote vehicle dynamics modeling, simulation, and associated technologies for the purpose of aerospace system development, analysis, and training. The committee emphasizes simulation of flight vehicles, however it includes missiles and other dynamic vehicles.

The technical committee has developed a standard for flight dynamics systems modeling. The initial purpose of the standard is to define data formats for function table and time history data, standardized axis systems, standardized variable naming conventions, and to provide electronic means for transmittal of models in a standard format. This standard is presented with the rational for the components of the standard and the decisions made in adopting the standard. The process for making this standard an ANSI and ISO standard is also briefly reviewed.

The Modeling and Simulation Technical Committee has a goal of being a key organization in flight vehicle dynamics modeling standards and the certified models that result from such standards.

Objective of the Standard

Standard Method for the Transfer of Air Vehicle Modeling Data

There are many different levels of standards that could be produced to support the development of flight vehicle dynamics models. This paper specifically addresses the first logical step in the development of these standards, a standard for the exchange of simulation modeling data (Figure 1). How this standard would benefit the modeling and simulation community will be explained below.

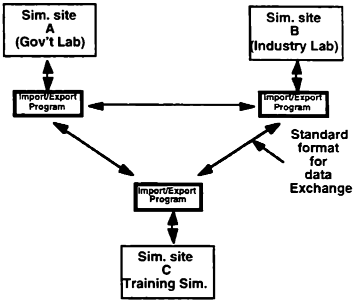


Figure 1: The Objective of This Standard

Presently throughout industry there exists many versions of the same vehicle dynamic models. These models are used both for analysis, RDT&E simulations, and training simulations. For example, there are over ten F/A-18 real-time man-in-the-loop simulations of models A through D alone. There are many more analytical models. These models are at government, industry, academia, and even foreign countries such as Canada and Australia. Since there is no standard data format for these models, once a model is sent from one organization to another it becomes a separate version and any developments of one organization are lost to the other. Loss in synergism between the ten plus organizations working actively with F-18 simulation is significant. The fidelity of each model is uncertain and the verification, validation, and certification is virtually non-existent.

A cohesive data format for the exchange of information between facilities would dramatically alleviate this problem. Work with the AIAA has emphasized that this standard would be eminently acceptable to government and industry because it is a format for exchange of information and is not a standard for the internal use of the information. It is believed and accepted that it will lead over time to organizations evolving into using the information the same way, but immediately allows a quick import/export concept to be implemented at each facility. In other words, to use the standard, the facility only needs an import utility program to change the standard format to their internal

* Sr. Member AIAA

format (Figure 1). Likewise, to send data to another organization they need only to write an export utility to take it from their internal format and write it to the standard. Over time each organization would logically evolve to using the standard as their own internal format. However, they would not be forced to do so on a specific schedule or to qualify for any specific program. This makes the standard very attractive to all organizations.

Later Potential for More Specific Standards

The data exchange format standard is also a logical base for future specific standards. There is a logical progression for standards of this type (Figure 2). To have a standard software program you must first define the framework of the model (the axis system, for instance) and agree on the input and output functionality of each subprogram. Before you standardize these items, the lower level items must be agreed upon, such as table lookups and variable names. The variable name definitions are required in order to send information from one facility to another in a concise well-defined manner. For example, if you are going to transfer an aerodynamic function from facility A to facility B, that function (dependent variable) is a function of independent variables. The independent variables must be clearly defined to define the dependent variable. This requires the standard to include some standard variable definitions. In addition, a time history data format is required to verify that a model received was implemented properly. There must be verification data sent, so facility B can make the same test runs that facility A performed and verify that both simulations perform the same (not necessarily like the aircraft).

Therefore, the first logical step in defining more specific standards for vehicle modeling is the definition of:

⇒ function table formats (for transfer of data)
⇒ time history formats (for verification)
⇒ variable names (for defining the information sent)
⇒ axis systems (for defining the variables sent).

Later standards would logically be developed based on this standard.

The logical development of simulation standards is illustrated in Figure 2.

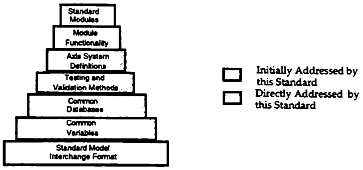

Standard Modules
Module Functionality
Axis System Definitions
Testing and Validation Methods
Common Databases
Common Variables
Standard Model Interchange Format
□ Initially Addressed by this Standard
□ Directly Addressed by this Standard

Figure 2: The Logical Progress of Dynamic Modeling Standard Development

Status of the Standard

AIAA Modeling Simulation Technical Committee Work Completed to Date

Over approximately the last three years the M&S Technical Committee has worked on this draft standard on a volunteer basis [1]. It consists of four sections and an introduction.

The four sections are:

⇒ Standard Variable Names
⇒ Standard Axis Systems
⇒ Standard Function Table
⇒ Standard Time History Data Formats

This work has attempted to draw upon other existing work. For example, the axis system is consistent with published AIAA recommended practices [2] and DIS axis system definitions [3]. The variable system naming is patterned after the STARS Program [4] and NASA Ames Research Center [5] simulation software.

With the exception of minor finishing touches the variable naming convention and axis system definition work is complete.

Standard Variable Names

The variable naming convention includes a methodology for naming new variables to allow consistency in future extensions of the variable name standards. It is also consistent with object oriented design techniques and structured software.

The variable name standard actually has two distinct parts:

- A list of standard variable names for variables typical of aircraft simulations
- A methodology and convention for defining variables specific to a simulation domain so the

list may be expanded at the national and local organizational levels.

This section will discuss the convention and philosophy used for naming of simulation variables. The purpose of this is so when other variables are added to the list they will follow the same general convention.

General rules for naming variables:

- Variables shall have meaningful names. Mnemonics will not be used. Standard abbreviations are allowed.
- Distinct word in variable names shall be separated by underscores.
- Variable names shall not exceed 60 characters in length. Brief but concise names are most effective.
- A short variable name (8 characters) is allowed as an alias.
- The first letter of each word will be capitalized and the remaining lower case. This improves readability[6]. Abbreviations are all capitals. Units should be all lower case. This reduces ambiguity.

Methodology for Creating New Names

The method for creating a new variable name is as follows.

Each name has up to six components. All components are not required to be used because in many cases they do not apply. These components are:

(prefix)_(variable source domain)_(axis or reference system)_(specific axis or reference)_(core name)_(units)

Prefix is Key Component

The purpose of the prefix is to identify the variable in two ways. The prefix is used to denote the key variables in the model, which are the model states, state derivatives, and inputs. Mathematically, all control over the model and outputs of the model are derived from these. The prefix also is used to identify different models in a multi-model simulation.

It is important to emphasize identification of the states and the inputs are a key factor in simulator software development and maintenance of existing simulation software. This convention should hold true for control states, landing gear states, and any dynamic system in the vehicle.

The only remaining decision to make in the variable naming convention is the distinctions between inputs to the simulation versus inputs to the dynamic model of the simulation. For example, an input to the simulation might be the initial altitude to start the simulation. This is not necessarily an input to the dynamic model, but rather a condition for initialization.

Variable Source Domain

This is the object domain in which the variable is used. In object oriented design, it would logically be the object. The parameter source domain is normally not included if it (or the object) is the vehicle or aircraft being simulated (ex. airspeed.). User defined variable names should normally include the domain name.

Some domain examples:

Aero or Aerodynamic
Engine or Thrust
Controls
Wheel
Landing_Gear or rt_main_gear
Hydraulics or Primary_hydraulic

Variable name examples:

```
Aero_X_Component_Force_Coefficient
Thrust_Y_Component_Force_lbf
```

Axis or Reference System

This is the axis or reference system to which the variable is referenced. The standard axis system abbreviations are used. If no axis system pertains to the core variable name (ex. airspeed), then this part of the variable name may be omitted and there is no ambiguity.

Specific Axis or Reference

This is the specific axis or reference used within the axis system. If the axis or reference system component of the name is included in the name, then the specific axis or reference should also be included.

Core Variable Name

This is the most specific (hence core) name for the variable. All variable names must include this component of the name.

Core variable name examples:

```
Cosine_Of_Angle_Of_Sideslip
Rate
Turbulence_Velocity
Velocity
```

Suffix

The suffix is used to describe the units of the variable.

The convention for the suffix is simple and is followed for all variables. This will allow the user, the programmer, and the reader of the code to check for homogeneity of the units and is obviously self-documenting in this respect. Therefore, the units will be put on all variables except variables that are non-dimensional which will have no suffix. This also has the other significant advantage of making this standard consistent and acceptable in countries with the international system of units. For example, airspeed is just as acceptable as a standard both for the American system of units and the International system of units. In one case it's

`airspeed_f_p_s` for feet per second (f/s)

and in the other case

`airspeed_m_p_s` for meters per second (m/s)

Examples:

`s_Body_X_Velocity_f_p_s`

(s_ prefix indicates that this variable is a state)

`sd_Body_X_Acceleration_f_p_s2` (sd_ prefix indicates that this variable is a state derivative)

`F2_s_Body_X_Velocity_f_p_s`

or alternatively

`s_Body_X_Velocity_f_p_s(F18_001)`

or alternatively

`F18_001_s_Body_X_Velocity_f_p_s`

(F2_ prefix indicates that this variable of the #2 friendly aircraft or vehicle, alternatively identified as F18_001. Note that the s_ prefix indicates that this is also a state variable).

`T1_true_airspeed_f_p_s` (T1_ prefix indicates that this is the `"true_airspeed_f_p_s"` variable of the #1 threat aircraft or vehicle)

`Body_Roll_Rate_d_p_s` (The lack of the s_ prefix indicates that this is not the roll rate state, but an output.)

Standard Axis System Definition

The axis system definition is complete and based on AIAA recommended practice [2] and DIS standards [3] already published. The standard is the overlap of both of these standards. The variable names reference the axis systems used.

Axis system standards also are reflected in the variable naming convention. When applicable, the axis system is included in the variable name. The following systems are proposed.

Geocentric Earth Fixed-Axis System

The Geocentric Earth Fixed-Axis system is a system with both the origin and axis fixed relative to, and rotating with, the earth. The origin is at the center of the earth, the x_G axis being the continuation of the line from the center of the earth through the intersection of the Greenwich Meridian and the Equator, the z_G axis being the mean spin axis of the earth, positive the north, and the y_G axis completing the right hand triad.

Body Axis Coordinate System

It is a system fixed in the vehicle with the origin at the center of mass consisting of the following axis orientation. The x axis is in the reference plane of the vehicle, or if the origin is outside that plane, is in that plane through the origin that is parallel to the reference plane, and positive forward. The reference plane is the plane of symmetry or clearly specified alternative. The y axis is normal to the reference plane and positive to the right. The z axis lies in or is parallel to the reference plane and completes the orthogonal right hand triad.

Flat Earth Axis System

The third axis system is defined only for convenience on creating verification data. It is a fixed, non-rotating, flat earth with no mapping to a round earth coordinate system, therefore, latitude and longitude are meaningless. The purpose of this coordinate system is to allow, if desired, vehicle checkout simulation to be performed in this axis system. This simplifies the use of this standard by the simulation facilities which do not normally use a round or oblate spheroid, rotating earth model.

The Flat Earth reference system is situated on the earth's surface directly under the cg of the vehicle at the initialization of the simulation. The x axis on the local frame points northwards and the y axis points eastward, with the z axis down. The x and y axis are parallel to the plane of the flat earth. This is a system adopted from NASA Ames [5].

Work in Progress and an Example

There are four subsections in progress, they are:

- ⇒ Time History Data Format
- ⇒ Function Table Data Format
- ⇒ Selection of the actual method of data transfer
- ⇒ Verification that data transfer was performed properly

The function table data and time history data formats are largely complete. The time history data format is actually taken from a flight test data format recently developed.

Function Table Format

A simple mathematical function is shown below as an example (Figure 3). The proposed format for this standard is a significant advance in that it includes data reference and confidence interval information for each data point. The reference documents the source of each data point and the confidence interval documents the statistical accuracy of the data point. A data format without this information would result in loss of what data was changed in the table over time (the reference documents this) and how to combine the information in the table with another source of information, such as another similar data table (the confidence intervals provide this).

Figure 3 presents a fairly standard three-dimensional set of data as is typical of aerodynamic data from flight test or from a wind tunnel. In the example given, lift coefficient is a function of angle-of-attack, Mach number, and a control position. Examination of the data will show the following characteristics for which a standard function table format must be able to satisfy. Close examination of the data given will show the following characteristics:

- The number of breakpoints of the independent variables varies for each independent variable. Not only are there a different number of angle-of-attack breakpoints for different functions, but different number of Mach number breakpoints.
- The values of the independent variable breakpoints are different.
- The valid ranges of the independent variables are different.
- The above three differences are not consistent for all the data. For example, in the example table the angle-of-attack breakpoints for Mach=0.6 and 0.7 for delta_s= -5 are identical.

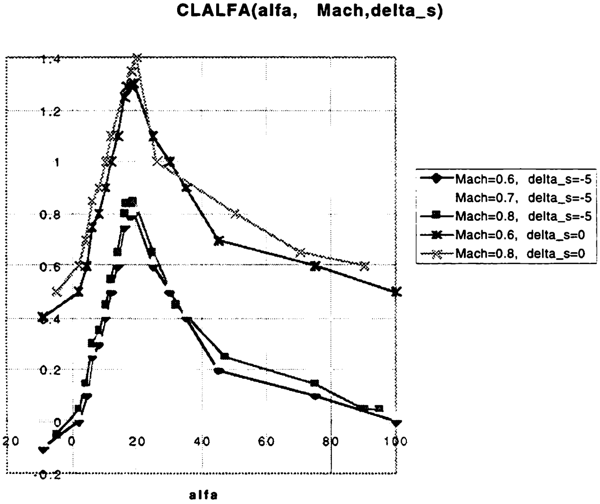


Figure 3. A Typical Non-linear Function

For the example data there is other information that is of significant importance to the user, without which the data is not very useful. In general this information is, where did the data come from, how is it to be used, what are the units of the dependent and independent variables, and what is the sign convention of the dependent and independent variables? For example, is the control position positive trailing edge up or trailing edge down? Exactly which control surface is it? Who created the table? How has it been modified and for what reason?

Mathematically there is also different additional information about the table that is not presented in the plot. Most significantly is, how accurate is the data estimated to be? Or, mathematically what is the confidence interval of the data? By what method is the data intended to be interpolated? For example, linear interpolation or by cubic spline interpolation? By what method is the data intended to be extrapolated for data with different ranges?

The standard data format has data elements that contain all of the above information. It is easy to be read by the human and the computer and has two parts:

- A header which documents the table
- A data section which contains the function data.

The header is illustrated as an example in Figure 4. In general, it is self documenting but a few further remarks are warranted. It has three parts, the table definition, the references used when the data was created and personnel involved in the creation, and the modification history of the table. The table definition includes the methods by which the table is meant to be interpolated; if data is meant to be extrapolated when the independent variable arguments exceed the limits of the table; the method to use to extrapolate; and how far to extrapolate (the limits that are to be placed on the independent variables, if any).

The data section of the table is shown in Figure 5. As in the header, the data section is intended to be self documenting and includes:

- The function data—the values of the dependent variable
- The breakpoint data—the values of the independent variables
- A reference for each data point
- A plus and minus 95% confidence for each data point.

The importance of the reference and confidence intervals for each data point cannot be overstated. The functional data is the heart of a model, it is what makes one non-linear model (airplane) different from another. The modeling and simulation industry in general has weak configuration management and tracking of this important information. This makes use of the reference critical.

The confidence interval (when known) is also important because it gives an intuitive and mathematical description of the estimated accuracy of the data. If the analyst has two estimates for the same data point, and these estimates have a confidence interval they can be mathematically combined and a new confidence interval established. The positive and negative values allow non-Gaussian (or non-normal) confidence intervals to be included. Entering only a positive value in the table indicates a Gaussian probability distribution for the data. Use of other probability distributions must be documented in the header.

Additional work that should be performed is to assure that these data formats are compliant with the DoD data dictionary system.

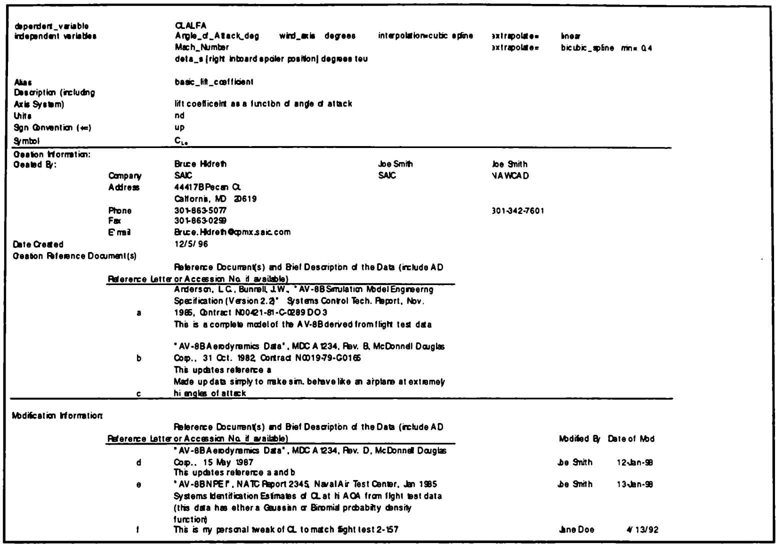
Time History Data Format

As for all the components of the standard adopted, the time history data format is taken from another project or standard that was recently developed for the collection and storage of time history data from actual flight or ground tests [7]. This data format was intended to capture all the pertinent data for actual flight tests and be suitable for synchronous or asynchronous systems with non-deterministic acquisition times. As in the case of the function table data, this has two parts to each data set—a flight record header and a time history data set.

The flight record contains information about the time history data that is included. It contains the typical information which would not change for a complete set of simulation runs and documents the facility that the run was made at, the times the simulation runs were made, and any particular configuration information so the data may be used in the future.

Time history data is just sets of ordered pairs. Each time history vector data contains the actual data for each run in a separate file and there will be a directory for each record (defined here as data on through data off). In that directory will be one file for each time history. For example, if you recorded angle-of-attack and angle-of-sideslip only, there would be one file for angle-of-attack and one file for angle-of-sideslip and each file would have time in it so if they were collected from different CPUs or different tasks running at different rates, each file would have a different time base.

The flight record data set and time history vector data sets are illustrated in Figure 6 and will be described in more detail below.



dependent_variable		CLALFA			
independent variables		Angle_of_Attack_deg wind_axis degrees	interpolation=cubic spline	extrapolate=	linear
		Mach_Number		extrapolate=	bicubic_spline min= 0.4
		deta_s (right inboard spoiler position) degrees teu			
Alias		basic_lift_coefficient			
Description (including					
Axis System)		lift coefficeint as a function of angle of attack			
Units		nd			
Sign Convention (+=)		up			
Symbol		$C_{L\alpha}$			
Creation Information:					
Created By:		Bruce Hildreth	Joe Smith	Joe Smith	
	Company	SAIC	SAIC	NAWCAD	
	Address	44417B Pecan Ct. California, MD 20619			
	Phone	301-863-5077		301-342-7601	
	Fax	301-863-0299			
	E mail	Bruce.Hildreth@cpmx.saic.com			
Date Created		12/5/96			
Creation Reference Document(s)					
	Reference Letter	or Accession No. if available) Reference Document(s) and Brief Description of the Data (include AD			
	a	Anderson, L.C., Bunnell, J.W., "AV-8B Simulation Model Engineering Specification (Version 2.2)" Systems Control Tech. Report, Nov. 1985, Contract N00421-81-C-0289 DO 3 This is a complete model of the AV-8B derived from flight test data			
	b	"AV-8B Aerodynamics Data", MDC A 1234, Rev. B, McDonnell Douglas Corp., 31 Oct. 1982, Contract N0019-79-G0165 This updates reference a			
	c	Made up data simply to make sim. behave like an airplane at extremely hi angles of attack			
Modification Information					
	Reference Letter	or Accession No. if available) Reference Document(s) and Brief Description of the Data (include AD		Modified By	Date of Mod
	d	"AV-8B Aerodynamics Data", MDC A 1234, Rev. D, McDonnell Douglas Corp., 15 May 1987 This updates reference a and b		Joe Smith	12-Jan-98
	e	"AV-8B NPE I", NATC Report 2345, Naval Air Test Center, Jan 1985 Systems Identification Estimates of CL at hi AOA from flight test data (this data has either a Gaussian or Binomial probabilty density function)		Joe Smith	13-Jan-98
	f	This is my personal tweak of CL to match flight test 2-157		Jane Doe	4/13/92

Figure 4. An Example of the Standard Header for a Function Table

APIs Available for this Database System

While the standard is not requiring a standard database, another advantage to this time history standard is there is a fairly simple UNIX and NT database system available for it with application programming interfaces (APIs) to allow the user to easily get access and store data in this database. It also allows APIs for plotting of the data. This software system is up and running and used on a routine basis at NAWCAD and SAIC.

Flight Record

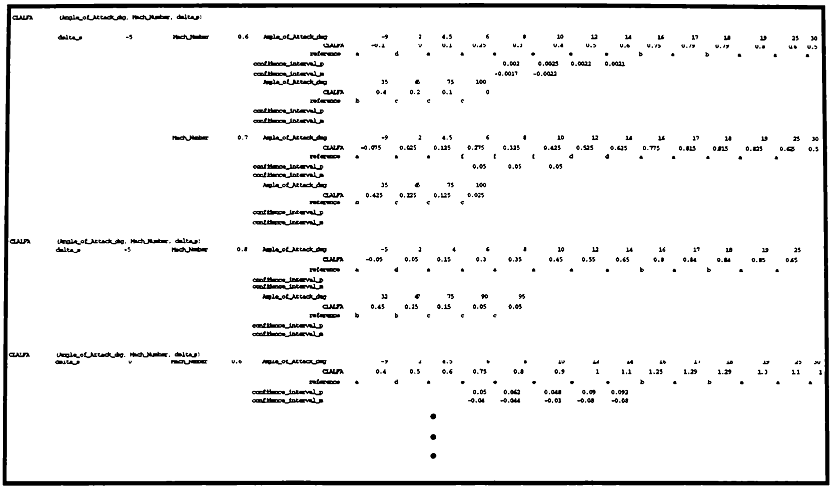
The main division of flight data in the standard format is by flight records. A *flight record* is defined as a complete set of test data and related header information covering a portion of a flight. If test data are continuously recorded for the duration of a flight, they are provided as one record. If data recording devices were turned off and on during a test, the data are stored as several flight records each representing the time between turning the recording system on and off.

Table 1 shows information contained in each flight record header file. The actual computer format will be text key word based so the file can contain as many key words as desired so if additional information is required new key words can be defined to be added to that database.

A key component is the test configuration file. This configuration file is the file name for an initial condition or test loading file for the simulator. Most simulators have one or more files used to configure the simulator for a standard set of runs. This is a key specification for simulator test data to know what the exact configuration of the simulation was for each record.

Time History Vector

The standard data stream stored is the time history vector. A *time history vector* is a single channel of recorded flight data along with its associated time tag. Conceptually, a time history vector is a N by 2 matrix with time points in the first column and data points in the second column. Each row contains the time and data value of a single measurement. Individual channels are stored in separate time history vectors. Since each channel is stored separately with its own time tags, there is no requirement for uniform sampling of data. Data should only be stored when sampled; there is no need to sample-and-hold or interpolate data to achieve uniform sampling. Also, in the case of a typical data drop out, neither the time nor the data value is included in the time history vector.



```
CLALFA      (Angle_of_Attack_deg, Mach_Number, delta_s)

    delta_s        -5        Mach_Number      0.6   Angle_of_Attack_deg        -9       2     4.5       6       8      10      12      14      16      17      18      19      25   30
                                                                  CLALFA     -0.1       0     0.1    0.25     0.3     0.4     0.5     0.6    0.75    0.79    0.79     0.8     0.6  0.5
                                                               reference   a       d       a       a       a       a       a       a       b       a       b       a       a       a
                                                 confidence_interval_p                                    0.002  0.0025  0.0022  0.0021
                                                 confidence_interval_m                                  -0.0017 -0.0022
                                                    Angle_of_Attack_deg        35      45      75     100
                                                                  CLALFA      0.4     0.2     0.1       0
                                                               reference   b       c       c       c
                                                 confidence_interval_p
                                                 confidence_interval_m

                             Mach_Number      0.7   Angle_of_Attack_deg        -9       2     4.5       6       8      10      12      14      16      17      18      19      25   30
                                                                  CLALFA   -0.075   0.025   0.125   0.275   0.325   0.425   0.525   0.625   0.775   0.815   0.815   0.825   0.625  0.5
                                                               reference   a       a       a       f       f       f       d       d       a       a       a       a       a
                                                 confidence_interval_p                            0.05    0.05    0.05
                                                 confidence_interval_m
                                                    Angle_of_Attack_deg        35      45      75     100
                                                                  CLALFA    0.425   0.225   0.125   0.025
                                                               reference   b       c       c       c
                                                 confidence_interval_p
                                                 confidence_interval_m

CLALFA      (Angle_of_Attack_deg, Mach_Number, delta_s)
    delta_s        -5        Mach_Number      0.8   Angle_of_Attack_deg        -5       2       4       6       8      10      12      14      16      17      18      19      25
                                                                  CLALFA    -0.05    0.05    0.15     0.3    0.35    0.45    0.55    0.65     0.8    0.84    0.84    0.85    0.65
                                                               reference   a       d       a       a       a       a       a       a       b       a       b       a       a
                                                 confidence_interval_p
                                                 confidence_interval_m
                                                    Angle_of_Attack_deg        32      40      75      90      95
                                                                  CLALFA     0.45    0.25    0.15    0.05    0.05
                                                               reference   b       b       c       c       c
                                                 confidence_interval_p
                                                 confidence_interval_m

CLALFA      (Angle_of_Attack_deg, Mach_Number, delta_s)
    delta_s         0        Mach_Number      0.9   Angle_of_Attack_deg        -9       2     4.5       6       8      10      12      14      16      17      18      19      25   30
                                                                  CLALFA      0.4     0.5     0.6    0.75     0.8     0.9       1     1.1    1.25    1.29    1.29     1.3     1.1    1
                                                               reference   a       d       a       a       a       a       a       a       b       a       b       a       a       a
                                                 confidence_interval_p                    0.05   0.062   0.048    0.09   0.092
                                                 confidence_interval_m                   -0.04  -0.044   -0.03   -0.08   -0.08
                                                                      •
                                                                      •
                                                                      •
```

Figure 5. An Example of the Data Section of a Standard Function Table

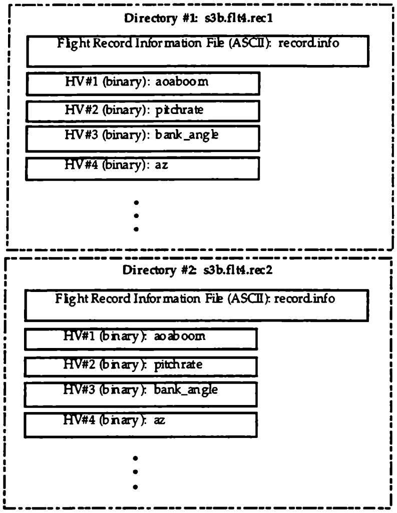
Table 1: Flight Record Information

Key	Brief Description
AIRCRAFT TYPE	Aircraft type
SIMULATION DESIGNATION	Aircraft bureau number
REFERENCE TIME	Reference time
FLIGHT NUMBER	Flight number
DESCRIPTION	Description of flight
CREW	Crew member name(s)
SIM OPERATORS	Simulator operator name(s)
RECORD START TIME	Record start time
RECORD STOP TIME	Record stop time
COMMENT	Misc. comments
REFERENCE WEIGHT	Reference weight
FUEL LOADING	Fuel loading - Frozen?
A/C LOAD CONFIG	Aircraft loading configuration
TEST CONFIG COMMENTS	Comments on the test configuration
TESTED CONFIG FILE	Simulator Initialization File(s)
NUMBER OF HV	Number of time history vectors
HV	Time history vector specification

Verification

The verification section of the standard remains to be completed. At this level of the standard the purpose of the verification is to assure that the simulation model is properly transferred from one site to another. Therefore the verification assures that one simulation performs like the other simulation, and does not assure that the simulations perform like the vehicle being simulated (often called validation).

The verification section of the standard will comprise of a set of test inputs and time history data recording the inputs, states, and selected outputs, designed to allow one simulation facility to easily duplicate a set of tests performed at another simulation facility. Of course, it will use variables, axis systems, and time history data formats from this standard to fully and unambiguously specify the tests to be performed, especially the initial conditions and control inputs for each test.



Directory #1: s3b.flt4.rec1
Flight Record Information File (ASCII): record.info
HV#1 (binary): aoaboom
HV#2 (binary): pitchrate
HV#3 (binary): bank_angle
HV#4 (binary): az
Directory #2: s3b.flt4.rec2
Flight Record Information File (ASCII): record.info
HV#1 (binary): aoaboom
HV#2 (binary): pitchrate
HV#3 (binary): bank_angle
HV#4 (binary): az

Figure 6: The Complete Structure of Two Records of Time History Data

The standard will include complex multi-axis control inputs to validate against. It will also include simple tests to assist in the debugging of problems.

Mechanization of the Standard Through HDF

The standard as discussed above only specifies the data that is to be included in the standard. There will also be a specification for the actual creation of files for electronic transmittal of a model in accordance with the standard.

The Hierarchical Data Format (HDF)[8] has been selected for this purpose. Specifically, the HDF Scientific Data application programming interface (SD API) is recommended for transferring multi-dimensional data tables and time history data. This interface is available in HDF version 4.0 release 1 and later versions.

HDF was chosen as the data transfer standard for a number of reasons. It is an existing format already in use in the scientific and engineering community. HDF was created at the National Center for Supercomputing Applications (NCSA) in 1988 and has developed into a comprehensive package. This package not only defines a data format but also includes libraries (in FORTRAN and C) for reading and writing HDF files. In addition, NCSA supplies command-line utilities for managing and viewing HDF files. Most importantly, this package is public-domain and freely accessible via the Internet.

HDF files are platform-independent. Files written on one computer can be read on another system with a different operating system. The FORTRAN and C libraries are available on a wide variety of commonly-used platforms. Machines that are currently supported include the Cray, HP, VAX, Sun, SGI, Macintosh, and IBM PC computers.

HDF files are self-describing. Each file contains information about itself that allows the HDF libraries to readily process the file without additional information. In addition, each file can contain annotations that improve the readability of the file. These annotations can include variable descriptions, units for each dimension, calibration data, and even file history information.

HDF is a versatile and extensible file format. There are various application programming interfaces that are already defined in the HDF documentation. The Scientific Data API is the most appropriate for simulation data. However, HDF files can also contain raster images, color palettes, multivariate table data, as well as groupings of any primary HDF object types. It is therefore expandable as this standard grows.

Future Developments

The committee has completed the framework and conceptual design of the standard with a significant amount of work on the actual standard accomplished. The technical decisions are made (preliminarily) for all the work. The AIAA has discussed the standard with DMSO and is striving to receive full DMSO support and recognition.

Activities that need to be completed are:

- Complete the verification section of the standard.
- Demonstrate its use at actual simulation facilities through the exchange of a model between several simulation facilities with different simulation systems. Development of unfinished application program interfaces (API) to support the standard would be part of this demonstration.
- Support the logistics of marketing the standard.

- Get customer feedback from the industry. This standard effort should perhaps become part of the Simulation Interoperability Standards Organization.
- Submit the standard through the AIAA as an ANSI and ISO standard.
- Finally, the committee would serve to support this standard and future standards. Just as hardware and software require life cycle support, so does a standard evolve as the needs of the user and technology change.

References

[1] B.L. Hildreth, *A Process for the Development of Simulation Standards*, AIAA Paper 94-3430.

[2] *Recommended Practice, Atmospheric and Space Flight Vehicle Coordinate Systems*, ANSI/AIAA, R-004-1992.

[3] *Standard for Distributed Interactive Simulation—Application Protocols, Version 2.0, Fourth Draft (Revised).* IST-CR-94-50, March 1994.

[4] *STARS Software Standards for the STARS Technology Demonstration Project*, The Boeing Company, Nov. 28, 1994.

[5] *A Standard Kinematic Model for Flight Simulation at NASA-Ames*, Richard E. McFarland, NASA CR-2497, Jan. 1975.

[6] David Straker, C-Style Standards and Guidelines, New York, Prentice Hall, 1992.

[7] Science Applications International Corporation, *Data Base Design Document for the Integrated Data Evaluation and Analysis System (IDEAS)*, SAIC Report No. 01-1393-2990-A002 (v. 1.1), Jan. 1998.

[8] *NCSA HDF Calling Interfaces and Utilities Version 3.1*, University of Illinois at Urbana-Champaign, http://www.ncsa.uiuc.edu/.

Ultimate Goal

The AIAA M&S Committee has the ultimate goal of being the designated point of focus for flight dynamics models standards. Not only for standards for simulation models, but for coordination of information exchange between models and coordination of development of standard flight dynamics software in the future.

MODEL SIMPLIFICATION USING STATISTICAL ANALYSIS: A CASE STUDY AND PROCESS

John C. Krainski
The Boeing Company
5000 E. McDowell Rd.
Mesa, Arizona 85201
john.krainski@boeing.com

Abstract

Real-time simulation requires simplified aerodynamic models because of limited computational resources. Traditional methods of building simplified models from empirical data and aerodynamic theory are costly and time-consuming. Statistical models, however, can be rapidly generated from experimental data using off-the-shelf software. Once a regression equation is obtained, estimated values for the dependent variable can be computed very easily without lengthy calculations and table lookups. This experiment consisted of replacing a traditional simplified model with a rapidly generated, statistically derived model, and evaluating it in a real-time flight simulation environment. It was found that a relatively simple, automatically generated, statistical model derived from experiments run on a high-fidelity model can drastically reduce computational requirements while maintaining significant accuracy. A simplified model of a General Electric T700-GE-701 turbine engine on an Apache AH-64A was generated using data obtained from test flights with a high-fidelity model. The simplified model is compared to the original in three areas: (1) quality of the fit to the collected data, (2) resources required to run the model, and (3) handling qualities in simulated flight. The results of this initial study indicate that using statistical analysis for model simplification is a useful technique for certain domains, such as a concept demonstrator on a platform with low computational power. In addition, a process for rapidly developing statistical models is presented (Appendix B).

Introduction

Traditional simplified models are typically generated by running experiments on a full-fidelity model or actual hardware and collecting data on selected independent variables. Values are collected under varying conditions, and the results are put in an n-dimensional table which defines a dependent variable y as a function of a set of n independent variables $\mathbf{x} = \{x_1, x_2, ..., x_n\}$. To find a value of y given $\mathbf{x}$, the model must find the table entry at each x_ith row. When the value of x_i falls between two known values, y must be interpolated. Frequently, this interpolation must occur in all n dimensions.

If the table lookup and interpolation algorithms are efficient, the simplified model will execute faster than a full-fidelity model. This method also has the advantage that, if the experiments are thorough and carefully designed, the simplified model will perform nearly the same as the full-fidelity model over a desired operating range. However, for complex models, where n is large and many interpolations are required, significant computational resources are still needed.

In some situations, it is desirable to have a simplified model that has much higher performance and lower fidelity. For instance, take the case of a concept demonstrator that must run on a laptop PC. The aircraft still needs to be flown in real-time, but the responses don't need to be as precise as they are in a handling qualities environment. It would also be desirable to obtain the simplified model quickly, perhaps with only minimal aid from a domain expert.

A statistically derived model can meet these needs. Multiple regression is a well-known statistical method for generating simplified models. It can be done automatically by commercial off-the-shelf software. Once an experiment is defined, a non-domain expert can collect data and generate the statistical model, with greatly reduced need for expert consultation. Finally, the resulting model is simply a polynomial function of the independent variables in $\mathbf{x}$, so it is easy to compute.

Multiple regression is used on experimental data to generate a function that approximates the dependent variable y in terms of n independent variables $x_1, x_2, ..., x_n$:

$$y = f(x_1, x_2, ..., x_n) \qquad (1)$$

A complete regression analysis includes the regression function f and a number of metrics to describe the quality of the fit to the original data. For a discussion of these metrics, please refer to Appendix A.

The Experiment

A handling qualities model of the Apache AH64-A was used for the test flight. This model, which contains two T700-GE-701 engines, is used daily by pilots to evaluate advanced avionics systems.

Engine output torque (hereafter referred to as *output torque* or *Q*) was selected as the dependent variable to model. Data was obtained from a simulated test flight that consisted of a number of maneuvers designed to exercise the engine performance envelope under normal flight conditions. Data was collected at a rate of 2 Hz on about 40 candidate predictors of output torque, which included pilot inputs, control surface actuator positions, and velocity and acceleration vectors. Statistical analysis software was then used to generate a second-order linear model of output torque using stepwise multiple regression (See Appendix A for details) on the collected data.

The resulting simplified model was of the form

$$Q_E = c_0 + c_1 x_1^{n_1} + c_2 x_2^{n_2} + \ldots + c_m x_m^{n_m} \qquad (2)$$

where Q_E is the estimated output torque, each c_i is a constant and each x_i is an independent variable. For a *k*th-order model, $n_i \in \{1,k\}$.

The Test Flight

Data collection took place during a test flight which consisted of a sequence of maneuvers designed to exercise the engine performance envelope under normal flight conditions. Extreme maneuvers, such as barrel rolls, steep turns, and quick stops were not performed in this test flight. The following is a list of the maneuvers performed:

- Hover at 100 feet
- Climb to 1000 feet
- Descent to original altitude
- Acceleration to cruise (~100 kts)
- 30 degree coordinated left turn, then 30 degree right turn
- 60 degree coordinated left turn, then 60 degree right turn
- Serpentine flight (up, down, up, down) at cruise speed
- From straight and level flight, acceleration to maximum airspeed (~150 kts)
- Deceleration to hover
- Lateral flight to the right at 40 kts, then laterally left at 40 kts
- 360 degree pedal turn left, then 360 degree pedal turn right

During the twelve minute flight, flown by a test pilot experienced with this flight model, the output torque varied from 0% to 120% of rated shaft torque. The variation in *Q* over the entire flight is shown as M_0 in Figure 1.

Data Collected

Data was collected on 42 candidate predictors of output torque at a rate of 2 Hz during the 12 minute flight, resulting in about 1400 data points. A candidate predictor is an independent variable that is believed to have a significant effect on *Q*. Selecting appropriate variables proved to be non-trivial; some knowledge of helicopter aerodynamics was necessary. Our simulator has hundreds of variables available for collection, but the statistics software had strict limitations on the number of variables it could handle, so careful selection of candidate predictors was a necessity. Therefore a domain expert was consulted for this task. The candidate predictors selected included pilot inputs, control surface actuator positions, and velocity and acceleration vectors. Below are the collected variables (vectors were counted as three variables):

- Cyclic and Collective Stick Positions, Pedal Position
- Main and Tail Rotor Actuator Positions, Stabilator Position
- Main Rotor Pitch, Disk Loading, and Normal Load Factor
- Body and Earth Relative Velocity and Acceleration Vectors
- Inertial Position, Velocity, and Acceleration Vectors
- Attitude Position, Rate, and Acceleration Vectors
- True Airspeed
- Ground Speed
- Radar Altitude Rate
- Total Vehicle Weight
- Barometric Pressure
- Fuel Flow
- Rotor RPM

Results

Four simplified models, M_1, M_2, M_3, and M_4 were generated from the data collected during the test flight of the high-fidelity model M_0. Three of the simplified models were tested in a simulator. Below are descriptions of each model:

M_1: This model was obtained by stepwise regression and contained 6 variables. It had poor fits in steady states such as cruise and hover, and during large roll maneuvers. However, it executed very quickly, about 700 times faster than M_0.

M_2: A hybrid model designed to combine M_1 with a specialized model that provided superior handling during roll maneuvers M_R and during zero-acceleration maneuvers M_Z. The roll maneuvers and zero acceleration maneuvers (hover and cruise) were isolated from the rest of the test flight, and M_R and M_Z were generated. M_2 displayed superior performance in these areas.

M_3: Using the same 6 variables as M_1,, a fifth-order model was created using standard multiple regression (not stepwise). This model fit better in the steady state and roll maneuvers, but executed much slower. It only ran 16 times faster than the high-fidelity model.

M_4: This is a second order model using multiple regression (again, not stepwise) on all forty-two variables. The result is second order equation with 84 variables. This had a higher R-squared than Models 1 and 2, but still had problems fitting the steady state and roll maneuvers. It ran about 15 times faster than M_0.

M_1 and M_2 will be discussed at length in the following sections because they provided the greatest performance gains. M_4 was not tested because of the technical complexity of implementing it. An estimate of its performance statistics using a mock version of the model are provided for reference in *Table 1*.

M_1 had an R-squared of 95.1%, and was a very good fit to M_0. Its performance in terms of speed was remarkable; it executed 708 times faster than M_0. Below are the predictors in M_1 and M_3. They were selected by the stepwise regression process as the variables that had the most significant effect on output torque Q. They are listed in order of their significance with the most significant first.

• Collective Actuator Position	ρ
• Tail Rotor Actuator Position	T
• Heading Rate of Change	$\dot{\psi}$
• Collective Stick Position	s
• Rotor RPM	Ω_R
• Pitch Acceleration	$\ddot{\theta}$

M_2 also included a specialized roll model M_R. The most significant variables for M_R were:

• Collective Actuator Position	ρ
• Rotor RPM	Ω_R
• Roll Acceleration	$\ddot{\phi}$
• Stabilator Position	E

Below is a summary of the models:

***M_1*:**

$$Q = c_0 + c_1\rho^2 + c_2\mathrm{T}^2 + c_3\dot{\psi} + c_4 s + c_5\Omega_R{}^2 + c_6\ddot{\theta} \quad (3)$$

***M_2*:** If Roll angle < 30 degrees then

Q = same as M_1

else

$$Q = c_0 + c_1\rho + c_2\Omega_R + c_3\ddot{\phi} + c_4 E \quad (4)$$

If horizontal acceleration < 3 m/s^2 then

$$Q = Q + 3\% \quad (5)$$

***M_3*:**

$$\begin{aligned} &Q = c_0 + c_1\rho + c_2\mathrm{T} + c_3\dot{\psi} + c_4 s + c_5\Omega_R + c_6\ddot{\theta} \\ &+ c_0 + c_1\rho^2 + c_2\mathrm{T}^2 + c_3\dot{\psi}^2 + c_4 s^2 + c_5\Omega_R{}^2 + c_6\ddot{\theta}^2 \\ &+ \ldots \\ &c_0 + c_1\rho^5 + c_2\mathrm{T}^5 + c_3\dot{\psi}^5 + c_4 s^5 + c_5\Omega_R{}^5 + c_6\ddot{\theta}^5 \end{aligned} \quad (6)$$

The simplified models implement equations (3), (4), (5), and (6), plus one additional statement of the form

```
if Engine_Output_Torque < zero then
    set Engine_Output_Torque to zero
```

This line was necessary because a negative value of output torque was invalid . Interestingly, the same logic also appeared in M_0.

Figure 1 shows M_1 superimposed on M_0. Note the boxed areas that show where M_1 fits poorly. Figure 2 illustrates a roll maneuver where M_1 fits poorly, and shows how M_2 corrects that problem with the specialized roll model M_R. Figure 3 illustrates how M_R only fits the roll maneuvers. Throughout the rest of the flight, M_R does not fit at all. Figure 4 shows M_3 superimposed on M_0.. The boxed areas illustrate improvement over M_1.

Performance Gains

Table 1 shows the performance gains in terms the average time to execute one iteration of the model. It also shows the lines of code required to implement the model and the size of the compiled object code. All values refer to the output torque calculation portion of the model only, and do not include any other components of the engine model. The figures for M_0 include "library" routines such as table lookup functions.

M_1 had the highest performance gain, running at 708 times the speed of M_0. It was followed closely by M_2 at 613 times the speed of M_0. M_3 paid a price for its greater accuracy; it ran only 16 times faster than M_0. M_4 ran at about the same speed as M_3.

Pilot Evaluation

M_1: The pilot was able to perform nearly all the maneuvers in the original test flight. However, it was observed that the engine model suffered from slow response time and provided insufficient power to conduct more demanding maneuvers. It also exhibited substandard performance during high bank coordinated turns and when there was zero acceleration, i.e., during cruise and hover.

The aircraft would not bank more than 70 degrees. Normally, this aircraft is capable of a barrel roll (360 degree roll). During hover, the aircraft tended to lose altitude very slowly because of insufficient power. Also, when accelerating from cruise, only a maximum of 135 knots could be achieved, which is well below the maximum speed of this aircraft.

M_2: The goal of this model was to correct the bank and zero acceleration problems of M_1. When the aircraft had more than 30 degrees of roll, a specialized roll model M_R dynamically took over. This resulted in improved performance during coordinated turn maneuvers. The pilot was able to achieve an 85 degree bank, but still could not perform a barrel roll.

When accelerating from cruise, the aircraft attained a speed of 164 knots, which is more realistic for this aircraft.

There were still some difficulties with handling the aircraft, such as slow response during extreme maneuvers. The pilot indicated that this model performed acceptably for normal flight conditions, but that some experience would be required to handle it well.

Analysis

M_1: The slow response time was a result of the model's poor fit at points where the slope of the graph changes rapidly. A higher-order model improved performance in these areas. Two maneuvers where insufficient power was observed were acceleration to maximum speed from cruise, in which case the aircraft could only achieve 135 kts when the true maximum speed should have been significantly higher, and a barrel roll, where the pilot could not achieve a bank angle of greater than 75 degrees.

The inability to accelerate to maximum speed is due to the "steady state" problem. Statistical models depend on changes in variables. When variables are not changing, it is difficult to fit a model to the data. The regression will interpret noise as variations, and generate a poor or even invalid model. This is the case during cruise: stick position is constant, velocity is constant, pitch is constant. Acceleration to maximum airspeed required very little change in stick position.

In executing barrel roll, there was no term in the equation to indicate roll rate or cyclic stick position, which made the model insensitive to the most significant independent variable in this maneuver. When roll was included, the model was not much better. This is the result of having few roll maneuvers in our test flight. When a hybrid model M_2 was created that had a special model for high banking, performance was significantly better.

M_0 uses Collective Stick position and Rotor RPM directly, so these variables were expected to be play a significant role. And they did; both appeared in the regression equation. Both the collective stick and main rotor actuator appear, with the actuator having the more significant effect on Q. This is because there is a damper/stability augmentation system between the collective stick and the actuator, so the actuator reflects the true main rotor blade pitch. Clearly, the tail rotor actuator and heading rate are coupled, and would be expected to have an effect on power required. An increase in tail rotor blade pitch increases the engine power requirement. This information is transmitted to the engine by a reduction in main rotor speed. Since the goal of the engine controller is to maintain constant main rotor RPM, this would cause an increased power requirement. Finally, a change in aircraft pitch should also have an effect on the power required. It should be noted that other variables may be expected to have a more significant effect on output torque than this simplified model indicates. This can be attributed to the multiple regression analysis which may not always identify the best subset of variables.

Originally, fuel flow was selected as a predictor, and initial statistical analysis showed it to be the best predictor of Q. Ultimately, however, fuel flow had to be removed from the model because it was actually dependent on output torque. This particular model computed the torque required, then signaled a corresponding change in fuel flow. Thus, the underlying mechanics of the model must be considered in the development of the simplified model.

M_R had two variables that did not appear in M_I, namely Roll Acceleration and Stabilator Position. It also contained the main rotor actuator position, but had very different coefficients from Model 1.

The following is a brief statistical analysis of M_I. These results are summarized in Table 2. The R-squared was 95.1%. The standard deviation of the residuals was 6.2, which indicates that 68% of all computed values were within 6.2 (% of rated shaft torque) of the actual value, 95% were within 12.4, and 99.7% were within 18.6. Finally, 93% of the standardized residuals were below 2.0 (a standardized residual below 2.0 is considered acceptable[1]).

Additional Models

Other models were developed, including ones with logarithmic and exponential terms, and terms with interactions between independent variables. However, none of these models proved to have a better fit than those listed above. It should be noted that because of limitations in the statistical software, a truly exhaustive model with interactions could not be created.

Conclusions

Using statistical methods for model simplification is a viable technique that enables rapid generation of reasonably accurate models that save significant computation resources. Although there are limitations to this method, it does provide for quick model generation, has limited need for a domain expert, and results in significantly reduced computational requirements. The models generated are appropriate for low-fidelity, high-speed environments, such as a laptop-based concept demonstrator.

There are, of course, disadvantages to this method. Since the model was generated over a series of maneuvers, it is possible that one or more maneuvers fit very poorly, while others fit very well. This was, in fact, observed in this experiment. The quality of the statistical model depends as much on the experiments performed as on the variables collected. In this case, a barrel roll was not included in the original experiment. In fact, there was not much rolling throughout the test flight. This accounts for the poor fit during roll maneuvers.

Another potential pitfall is that it is possible to generate a model that has little physical meaning if the experiment is designed incorrectly. For example, generating a simplified model using only the 60 degree right turn maneuver, resulted in a 91% fit using just X and Y positions. While this model might work well for this maneuver, it is obvious that this would be insufficient for most maneuvers, specifically anything resulting in a change in Z-position or attitude.

References

1. Neter, J. & Wasserman, W., *Applied Linear Statistical Models*, Richard D. Irwin, Inc. Homewood, Illinois, 1974.
2. Devore, Jay L., *Probability and Statistics for Engineering and the Sciences*, Brooks/Cole Publishing Company, Pacific Grove, California, 1991.
3. Stepniewski, W.Z. & Keys, C.N., *Rotary-Wing Aerodynamics*, Dover Publications, Inc., New York, 1984
4. Prouty, Raymond W., *Helicopter Performance, Stability, and Control,* PWS Publishers, Boston, Massachusetts, 1986.

Acknowledgments

I would like to thank the following people, without whose expertise this research could not have been completed: *Rick Esquibel* for help with statistical issues, *Lee Kralicek* for sharing his helicopter aerodynamics knowledge and experience with the model code, and *John Little* for his flying expertise.

Appendix A: Statistical Analysis Methodology & Metrics[1,2]

Stepwise multiple regression quickly identifies a useful set of predictors for a regression model, but not necessarily the best possible set. This is used when there are too many variables to evaluate every possible combination.

A *residual* is the difference between the computed value $\hat{Y}_i$ and actual value Y_i of the dependent variable. Thus, there is a residual for each row of data, and the smaller the residual, the better the fit.

A *standardized residual* is the residual divided by the standard deviation of the residuals. Standardized residuals have a variance of one. A standardized residual below 2.0 is regarded as acceptable[1].

The *R-Squared* is dependent on the residuals of each measurement, and the overall variability of the graph. R-Squared is defined by

$$R^2 = 1 - \frac{SSE}{SSTO}$$

where the *SSE* is the sum of squares of residuals

$$SSE = \sum((Y_i - \hat{Y}_i)^2)$$

and the SSTO is the sum of squares of deviations (the difference between the actual value Y_i and the mean $\overline{Y}$)

$$SSTO = \sum((Y_i - \overline{Y})^2)$$

Appendix B: A Process for Rapid Development of Simplified Models Using Statistical Analysis

The following is a step-by-step description of the process required to develop a simplified model using statistical analysis:

1. *Identify candidate predictors utilizing the original model code, if available, and domain expertise.*
2. *Design the experiment.* This step is clearly the most important. If the experiment does not cover the entire range of interest, the simplified model may be unacceptable in some situations. This step also requires domain expertise.
3. *Collect the data.* An automated tool was available that could collect the values of any variable in the model at any rate. Once the experiment was defined, data collection was trivial.
4. *Delete predictors that are constant throughout the experiment.* Among the variables that were collected, some did not change at all because they were not fully implemented, or the variations in the data were just random noise.
5. *Identify and delete invalid predictors that are actually dependent on the dependent variable.* In this case, fuel flow was identified as an invalid predictor. Variables that are highly correlated with the dependent variable should be suspect.
6. *Do correlation checks on every combination of predictors.* For every set of predictors that have a correlation of 1.0 (100%), delete all but one of the variables from the set. In this case, it was discovered that the same variable was collected from two different sources. Highly correlated variables caused problems for the statistical software.
7. *For a second order model, generate new columns for each predictor containing the squares.* Likewise for higher order models. Whether this step is required depends on the statistics software being used.
8. *For a model with interaction between predictors, generate new columns for each combination of predictors.* The number of combinations may be very large.
9. *Perform a stepwise regression on every predictor to generate the model.* The software performs this automatically.
10. *If there are any invalid or spurious predictors in the model, remove them from the data set and re-do stepwise regression.* This step requires a domain expert.
11. *Validate the new model by checking R-squared and residuals.* This step requires someone trained in statistics.
12. *Obtain the best fit possible, or the upper limit of the R-squared by generating a regression equation from every independent variable.* The resulting equation will be very large, but the R-squared will be the highest value that can be obtained with this data.
13. *Validate the model by using it in place of the original model and re-running the experiment.* In this case, the engine model was replaced and the original flight test was flown again.

Tables

Model	Lines of Code	Executable Size	Exec Time (microseconds)	Speed Increase over M_0
M_0	300	5072	184	n/a
M_1	8	120	0.26	708
M_2	15	136	0.30	613
M_3	8	4248	11.5	16
M_4	86	4264	12.27	15

Table 1: Model Performance Summary
This table shows the lines of code required to implement each model, the corresponding executable size, execution time and how many times faster than M0 each model executed.

Model	*Order*	*Vars*	*R^2*	σ^2	*Resid. < 2.0*
M_0	-	-	-	-	-
M_1	2	6	95.1	6.2	93.0
M_2	2	8	-	-	-
M_3	5	30	96.0	5.7	94.1
M_4	2	84	96.8	5.1	94.2

Table 2: Model Statistics Summary.
This table shows the order of the model, number of variables, R-squared, standard deviation, percent of standardized residuals below 2.0.

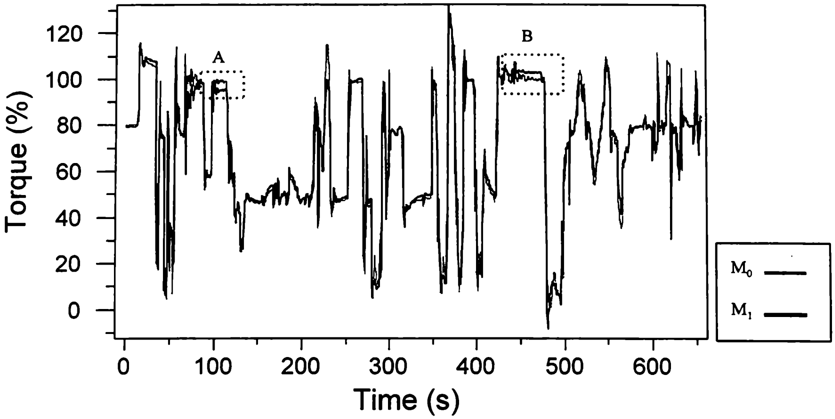


Figure 1: This graph shows the torque over the entire flight of M_0 (grey line) and M_1 (black line). The boxed areas are steady state manuevers where the model fit poorly.

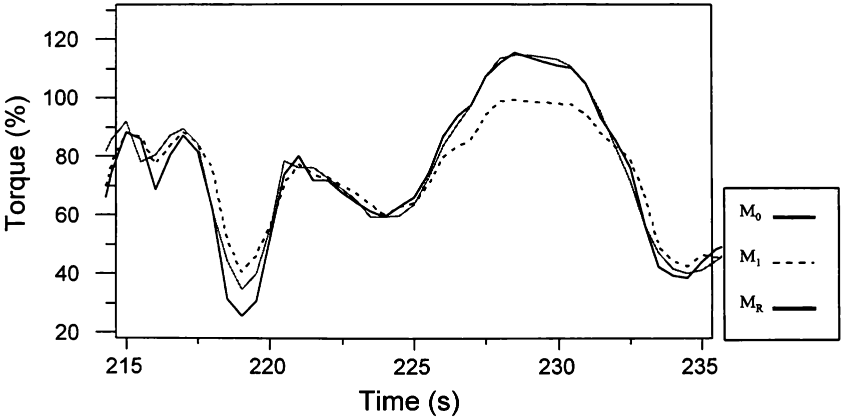


Figure 2: This graph shows M_0 (grey), the M_1 in (dashed line) and M_R (black) during a roll maneuver. Note the improved fit of M_R to M_0.

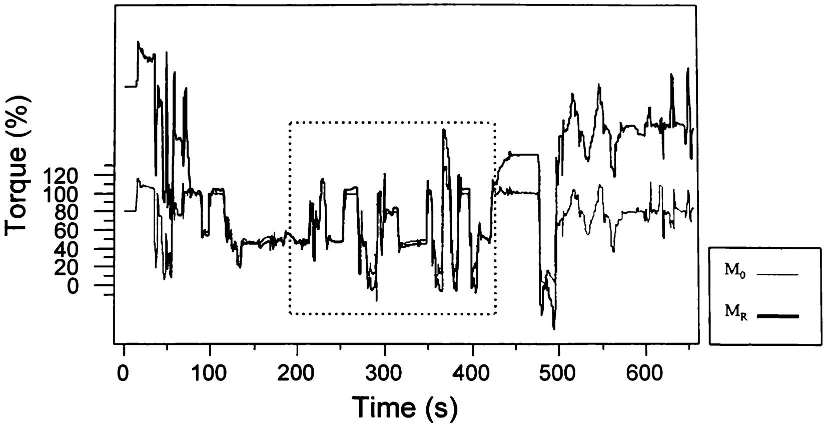


Figure 3: The specialized roll model M_R (black line) only fits well to M_0 (grey line) during roll maneuvers (boxed area), but does not fit at all during other maneuvers.

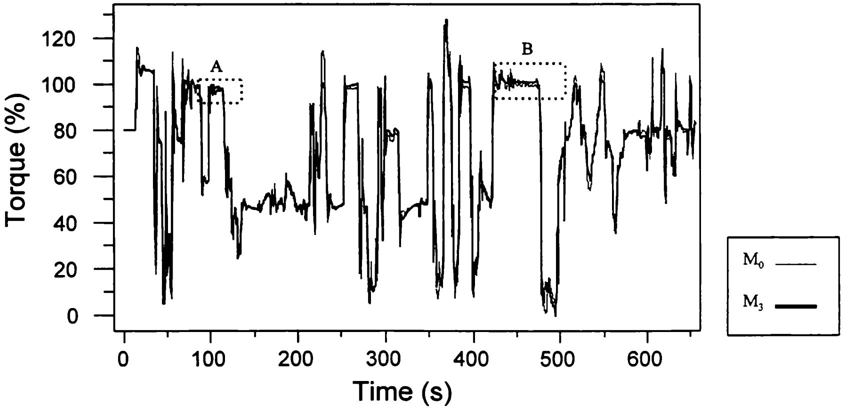


Figure 4: This is the graph of M_0 (grey line) and M_3 (black line). The boxed areas are where M_1 fit poorly (see *Figure 1*). Note that the M_3 fits much better here.

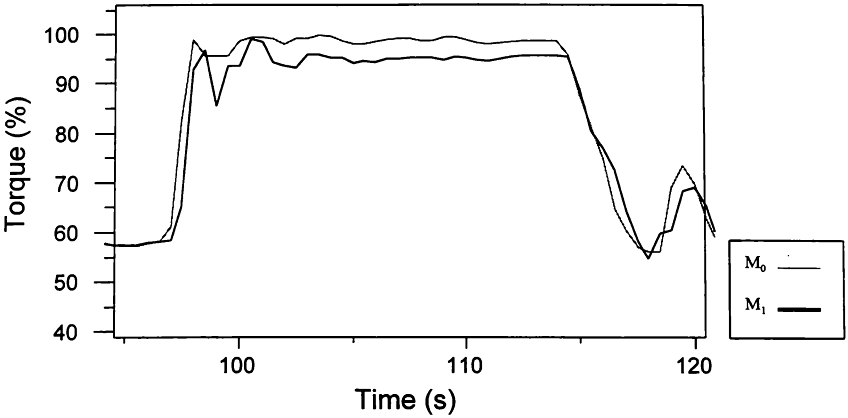


Figure 5: This graph shows a magnified view of boxed area *A* from *Figure 1*. Note the poor fits where there are rapid changes in slope. Also note the poor fit during a period of constant velocity from 100 to 115.

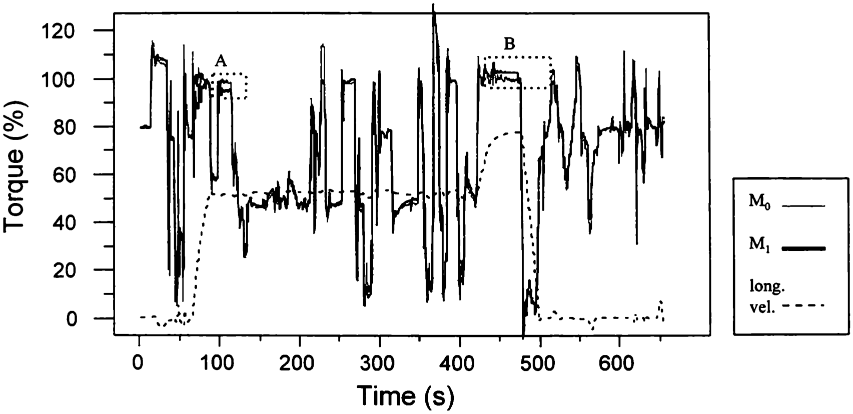


Figure 6: Illustration of the "steady state" problem. Superimposed over the actual torque (grey line) and M_1 (black line) is the longitudinal velocity (dashed line). Note that the worst fits occur when the aircraft has a constant velocity, specifically during hover and cruise.

AIAA-98-4357

Mathematical Modeling and Experimental Identification of a Model Helicopter

S. K. Kim* and D. M. Tilbury†
Department of Mechanical Engineering and Applied Mechanics
University of Michigan, Ann Arbor, MI 48109-2125
sungk@engin.umich.edu, tilbury@umich.edu

This paper presents a new mathematical model for a model-scale helicopter. Working from first principles and basic aerodynamics, the equations of motion for full six degree-of-freedom motion are derived. The control inputs considered are the four pilot commands from the radio transmitter: roll, pitch, yaw, and thrust. The model helicopter has a fast time-domain response due to its small size, and is inherently unstable. A flybar is used to augment the stability of a model helicopter to make it easier for a pilot to fly. The main contribution of this paper is to model the interaction between the flybar and the main rotor blade; it is shown how the flapping of the flybar increases the stability of the model helicopter as well as assists in its actuation. After the mathematical model is derived, some preliminary system identification experiments and results are presented. The paper ends with conclusions and a short description of future work.

Introduction

IT has been more than 20 years since the first commercial model helicopter was conceived, and since then, the design has significantly improved. Model helicopters are now well within the reach of many hobbyists and are also often used for commercial purposes, such as crop dusting or sport-event broadcasting. However, model helicopters are inherently unstable. Even with improved stability augmentation devices, a skilled, experienced pilot is required to control them during flight.

As a small, dynamically fast, unstable system, a model helicopter makes an excellent testbed for nonlinear control experiments. As a highly maneuverable machine, it also is an excellent testbed for path planning algorithms for autonomous robots. The integration of nonlinear control and path planning is our main interest in this project. As a preliminary step, this paper describes the new mathematical model that we have derived for a model helicopter control system, as well as some preliminary system identification experiments we have conducted. The outline of the paper is as follows. First, we briefly review some previous work on helicopter modeling and control. We then describe the new mathematical model that we have derived, working from first principles and basic aerodynamics. We then describe the system identification algorithm that we have used and present our preliminary results. We end with conclusions and a description of future work.

Previous Work

In the past several years, there have been a number of researchers interested in model helicopter control, and they have had various degree of success.

System Identification

A group at Caltech attempted to capture the main dynamic features of a model helicopter near hover with a linear time-invariant model. A MIMO identification algorithm was used to account for the significant cross-couplings in the model helicopter. Bendotti et al.[1] set up a model helicopter on a stand to provide three degrees of freedom: pitch, roll, and yaw rotations. Translational movements such as x, y, and z positions were neglected. The identification was carried out using a quadratic weighing factor with an iterative Gauss-Newton algorithm. The match between simulation and experiment was good in the pitch and roll directions, but was poor in the yaw direction due to the actuation asymmetry. Controllers were designed using H_∞ and LQG methods and the linear model that had been identified. Both controllers showed good response in disturbance rejection and command tracking; the H_∞ controller

*Ph.D. Student, University of Michigan, AIAA Student Member

†Assistant Professor, University of Michigan

had a faster response.

Another group from Caltech[2] did a similar identification and designed a LQR controller. The real-time control and data acquisition ran on the PC, which hosted the pulsewidth modulated IO board and Polhemus sensor board.[3] Again, the controller showed good performance only for pitch and roll response. Yaw response was poor for the same reason as above. Significant improvement on the yaw response was achieved when a separate loop shaping controller based on a lead-lag design was implemented.

Fuzzy Control

Dr. Sugeno from the Tokyo Institute of Technology[4] has had a considerable amount of success in flying the model helicopter for commercial purposes. The project's goal was to develop a controller for an unmanned helicopter that can operate under hostile conditions. The control system was designed using fuzzy control theory. The integrated control system, ranging from low level basic flight modes to high level supervisory control, takes a human voice as its input. Because human language voice commands are naturally imprecise or 'fuzzy,' the fuzzy logic framework was a good fit. The primary issue here was to design a controller that can easily include qualitative information as well as quantitative information. Some pre-experiment simulation was done on a Silicon Graphics IRIS workstation and a PC. The fuzzy controller rules were first constructed with the help of a actual human pilot's experience and knowledge. The rules were then tested and reformulated using the simulator. The helicopter was equipped with various sensors such as camera, gyroscope for 3D acceleration, Doppler speedometer, magnetic compass, laser altimeter, and GPS.

Unmanned Aerial Vehicle Competition

The annual unmanned aerial vehicle competition has been organizing a number of universities to complete a task of recognizing and moving an object to a designated target. So far, the competitors have typically been concentrating on the sensory issues such as the image processing, with a lesser emphasis on the control problems. Among the participating universities, USC used a concept called "behavior based control",[5,6] which implements a number of complex tasks with a collection of simple, interacting behaviors in parallel. Their approach seeks to use a behavioral based approach as a structure to unify navigation, motor control, and vision. This approach was inspired by the distributed yet unified control of biological systems. To maintain the safety of the helicopter, the lower-level behaviors may 'negotiate' with a mid-level to satisfy both sets of behavioral criteria. They claim this approach was useful in building an integrated control system; an open issue is the determination of a set of rules and principles for creating behaviors. The MIT, Boston University and Draper Lab team[7] was successful in building an autonomous model helicopter designed to hover, fly around, and recognize five randomly placed drums during the 1996 International Aerial Robotics Competition. The system consisted of the helicopter with various sensors such as GPS, IMU, altimeter, and compass, as well as a ground control station, a vision processor, and a safety pilot. The control system is divided into four closed loops.for roll, pitch, yaw, and collective/throttle with integrators to eliminate steady state errors. Pre-determined trim positions were used on those loops. They used relatively simple control laws to minimize the development time and be flexible to the changes in helicopter configuration. Again, their research effort was concentrated more on the sensor issues and the interactions between the various components than on the control structure itself.

Other UAV

The Naval Research Lab[8] has designed an airplane style UAV which is intended to be a missile decoy. It has an electric motor for a nose mounted propeller, three fiber-optic rate gyro sensors for the pitch, roll, and yaw axes, a barometric altimeter, and a pitot-static airspeed sensor. All the actuator dynamics are based on linear dynamic equations, and the single-axis control law is used for attitude and altitude control. From the launch through the cruise transition phase, the trajectory is based on a first-order linear fit to find the desired pitch angle profile based on 6DOF simulation. The reference pitch position is time-scheduled to follow the nominal trajectory. Various parameters are based on a wind tunnel experiment. Although the paper did not address the issues regarding how the aircraft would follow the pre-planned trajectory once it reached the cruise phase in satisfying the specific mission goal, it demonstrated the potential capability of the craft as a missile decoy.

In terms of UAV control, Kaminer et al.[9] pointed out that under a shifting wind disturbance, the traditional control scheme, in which the guidance and control parts are separate, may be inadequate to satisfy the precise tracking and frequent heading change necessary. To solve this problem, they examined a guidance and control scheme where the two are combined, so that the guidance law becomes an integral part of the feedback control system. With such design specifications as zero steady state error and certain bandwidth requirements, they devised a state-space coordinate system in which the linearization of the plant along a reference trajectory is time-invariant. This realization results in trajectories that consist of straight lines, arcs of circles of constant radii, and any combinations thereof. The control system itself is an LQR design. Their simulation showed a good result for an aircraft following a descending helical trajectory. However, the scheme only applies to certain specific trajectories as mentioned above, so it remains uncertain how useful the technique could be to a real system.

Sikorsky Aircraft[10] has been developing a saucer-type VTOL UAV called Cypher to meet various civil and military requirements. It has two counter-rotating, 4 ft long, coaxial, four-bladed main rotors shrouded by the 6.5 ft diameter main frame. Similar to a helicopter, it uses collective and cyclic pitch control for movement, powered by a 60 hp rotary engine. Sensors include radar, gyros, accelerometer, GPS, video camera, etc. To achieve a simple operator/vehicle interface, the operator is only required to send out basic maneuver commands such as takeoff, hover, cruise, desired heading, etc. Linear state space models are used to develop control laws and to determine specifications for servo and sensor bandwidths. The vehicle successfully demonstrated reconnaissance and surveillance capability, and is being improved to accomplish such tasks as mine-deploying and scouting missions. The vehicle is not yet completely autonomous nor is it capable of following a pre-determined trajectory.

Other research

At Purdue,[11] a student derived the dynamic equation of a model helicopter's vertical motion using blade element theory. For the experiment, a model helicopter was affixed on a stand to let it move only vertically. Uncertain parameters were estimated by interpolating the results from number of experiments with different parameters. After linearizing the dynamic model around the hover condition, controllers were designed using full state feedback pole-placement, LQR, and neural network techniques. All the controllers showed satisfactory performance.

Furuta et al.[12] derived a mathematical model of a model helicopter fixed on a stand, free to rotate around the pitch, roll, and yaw axis. Conservation of angular momentum was used to obtain the model. Realizing that the model helicopter is hard to control manually, they designed a stable tracking controller using the state space method. This stabilizing control input is then used to compare and validate the derived mathematical model with the experimental result. They realized that to be able to control the model helicopter in a full 6DOF situation, they need to expand the model to other degrees of freedom including the vertical motion. The model they derived was based on the full-scale helicopter modeling because they modeled a flapping rotor hub with spring, without explicitly taking the flybar into account.

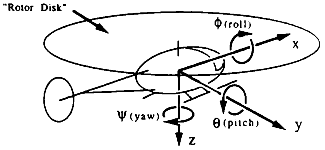


Fig. 1 The coordinates defined. The orientation variables, roll, pitch, yaw (ϕ, θ, ψ) and the position variables (x, y, z) are relative to the body frame (fixed to the helicopter).

Azuma[13] from University of Tokyo derived a mathematical model of a rigid rotor system. In this rotor system, each rotor blade is hinged but is spring loaded. He carefully derived a equation of motion using complex variables considering how different flapping stiffness affects the response of the system. Since this model does not take the dynamics of the helicopter as a whole, the validity of the model has not been fully proven.

Mathematical model

In this section, we will derive the dynamic equations of motion for the model helicopter including its actuator dynamics. We will use results from rigid body dynamics,[15] as well as basic aerodynamics and helicopter theory.[14,16]

Model vs. full scale helicopters

Before deriving the model helicopter dynamics, it is important to consider the differences between a full-scale helicopter and a model helicopter. First of all, a model helicopter has a much faster time-domain response due to its small size. Therefore, without employing an extra stability augmentation device, it would be extremely difficult for a human pilot to control it. A large control gyro with an airfoil, often referred to as a *flybar*, is almost always used nowadays to improve the stability characteristic around the pitch and roll axes and to minimize the actuator force required. Also, the tail rotor control for the model helicopter is assisted by an electronic gyro to further stabilize the yaw axis. Most full scale helicopters do not have such a control gyro on the rotor system. The large inertia of the rotor and fuselage and the flapping rotor hinge provide adequate stability.

Secondly, most model helicopters do not have a flapping hinge on the rotor to maximize the control power. Full scale helicopters often use either a free flapping rotor hinge or a spring-mounted hinge; these are usually absent on a model helicopter.

Rigid Body Equations

We will model the helicopter as a rigid body moving in space. As shown in Figure 1, we use the variables (x, y, z) to represent the position of the helicopter in body coordinates. We use the variables (ϕ, θ, ψ) to represent the roll, pitch, and yaw angles of the helicopter with respect to the body coordinates. Because it is a rigid body, the helicopter's position and orientation in body coordinates will always be zero; however, the velocity and acceleration expressions are greatly simplified by using these coordinates. We will assume the rotor system is completely rigid (there is no aeroelasticity effect), and that the the airfoil is symmetric and non-twisted. The aerodynamic interaction between the rotor and the fuselage is neglected. The aerodynamic expressions are based on 2-D analyses. This type of modeling is often called level-1 modeling and is appropriate for low bandwidth control and to observe the parametric trends for flying qualities and performance studies.[14]

The standard rigid body dynamical equation will be used to model the motion of the helicopter in its environment. The state vector $q = (x, y, z, \phi, \theta, \psi)^T$ contains the position and orientation information for the helicopter. Expressing the equation in coordinates gives[15]

$$\begin{bmatrix} mI_{3\times3} & 0 \\ 0 & \mathcal{I} \end{bmatrix} \ddot{q} + R_{IB}^T \dot{R}_{IB} \begin{bmatrix} mI_{3\times3} & 0 \\ 0 & \mathcal{I} \end{bmatrix} \dot{q} =$$

$$\begin{bmatrix} \begin{pmatrix} -D_{F_x} \\ -D_{F_y} \\ T - D_{F_z} \end{pmatrix} + R_{IB}^T \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} \\ M_\phi \\ M_\theta - T\ell_r \\ M_\psi + T_m - I_r\dot{\Omega} - K_g\dot{\psi} \end{bmatrix} \tag{1}$$

The rotation matrix R_{IB}^T transforming the inertial coordinates into body coordinates using yaw-pitch-roll (ZYX) Euler angles is given by

$$R_{IB} = e^{(\hat{z}\times)\psi} e^{(\hat{y}\times)\theta} e^{(\hat{x}\times)\phi}$$

$$= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

The cross "×" notation is used to represent the skew-symmetric cross-product matrix. For a vector $a = [a_1 \;\; a_2 \;\; a_3]^T$,

$$a\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

The rotational inertia matrix of the helicopter is given by

$$\mathcal{I} = \begin{pmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{pmatrix}$$

The terms on the right-hand side of the rigid body equation (1) include both the applied forces and disturbances. The D terms represent drag forces; these will be treated as disturbances in our model. The mass of the helicopter is given by m, and the fuselage inertias are I_{xx}, I_{yy}, I_{zz}. Terms such as I_{xy} and I_{yz} are zero due to the symmetry of the helicopter with respect to the x-z plane. Although I_{xz} is nonzero, because the helicopter is not symmetric with respect to the x-y plane, it is typically much smaller than the other terms. We have included it in the model for completeness. The rotor rotational inertia is I_r. The rotor angular velocity is Ω, and the offset between the rotor axis and the helicopter's center of gravity is ℓ_r. Usually this offset is expected to be zero for better handling quality. Nevertheless, we will assume this quantity is non-zero for generality. The gravitational acceleration constant is g. It is assumed that the helicopter's center of gravity is in-line with the rotor axis laterally.

The four independent inputs are T, the net thrust generated by the rotor, and M_ϕ, M_θ, M_ψ, the net moments acting on the helicopter, which are applied by a pilot. The mechanisms for creating these inputs will be described in the following section. The torque applied by the motor, T_m, is related to the thrust T and cannot be controlled independently. The electronic gyro acts as a damper on the yaw motion; we will use a simple linear model, $K_g\dot{\psi}$ for this gyro, although more sophisticated (PI controlled) gyros have recently become available.

Model helicopter actuation

A model helicopter moves forward when a pitching moment is first applied and the fuselage is tilted

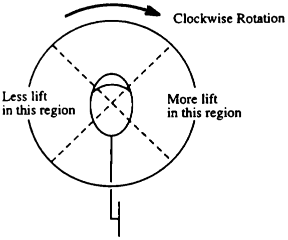


Fig. 2 The top view of the helicopter. The lift distribution on the rotor disk when a forward cyclic (pitch forward) input is applied. The precession effect will pitch the helicopter forward.

forward. The thrust vector T then gives a forward component for a forward thrust. A full scale helicopter only requires a forward tilt of the rotor disk to move forward while the fuselage stays level,[16] but this type of maneuver is not possible with a model helicopter.

There are four inputs available to the pilot of a model helicopter. These are physically controlled by two joysticks on the radio transmitter, each with two degrees of freedom. The left joystick commands throttle with collective pitch (up/down) and yaw (left/right), and the right joystick commands pitch cyclic (up/down) and roll cyclic (left/right). This is the most popular configuration used in the U.S. (type II*). The four values representing the positions of the sticks are encoded in a pulse-width modulated (PWM) signal, and sent via radio link to the helicopter. We will use these four positions $(\delta_t, \delta_\phi, \delta_\theta, \delta_\psi)$ as the inputs to our actuator dynamic equations because of the way we will control the helicopter later.

The throttle command (δ_t) controls the power to the main motor (T_m) as well as the collective pitch (θ_o) of the rotor blades. As the blade pitch increases, more lift is created, and the rotational motion of the main rotor blade is converted into vertical thrust. Usually, the relationship between θ_o and δ_t is linear $(\theta_o = K\delta_t$, where K is some constant). Meanwhile, an adequate torque T_m is applied to keep Ω constant. Sometimes, an electronic throttle governor is used for this purpose, but mostly a fixed simple smooth curve determining the necessary functional relationship between T_m and δ_t is programmed into the radio transmitter. This curve is traditionally obtained via trial and error. The yaw command (δ_ψ) controls the pitch of the tail rotor blade. The tail rotor on a helicopter is used to counteract the yaw moment created by the main rotor blade; thus, altering the amount of pitch on the tail rotor can create more or less total yaw moment for the helicopter. The pitch and roll commands influence the cyclic control, varying the cyclic pitch (θ_{cyc}) of the rotor blades around each cycle of rotation, creating different amounts of lift in different regions (as shown in Figure 2). These differing amounts of thrust create a moment around the rotor hub, and can thus create pitch and roll moments on the helicopter.

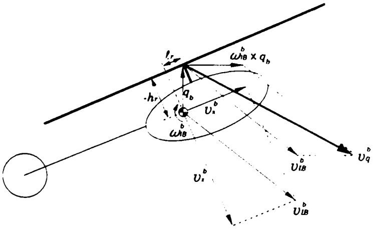


Fig. 3 Vector diagram explaining how the velocity v^b_q of the hub point q_b is calculated. The linear velocity of the helicopter CG is v^b_{IB} and its angular velocity is ω^b_{IB}. All velocities are measured in body coordinates. For clarity in the figure, lateral motions are neglected.

Before developing the dynamic equations, we introduce some basic aerodynamic terms that will be required. The advance ratio, μ, and the descent ratio, ν, represent the airspeed components parallel to and perpendicular to the rotor disk respectively.[17] They are close to zero when the helicopter is hovering. Both quantities are non-dimensionalized by $R\Omega$.[†] To define these quantities, we first need to find the velocity of the hub point with respect to the inertial coordinate frame represented in body coordinates. We denote this velocity by v^b_q. The angular velocity of the body frame as viewed in the body frame is ω^b_{IB}. The velocity of the CG relative to the inertial frame is v^b_{IB}. The coordinate of the rotor hub point is $q_b = [\ell_r \quad 0 \quad -h_r]^T$. The constants $-h_r$ and ℓ_r are the offsets of the rotor from the helicopter's center of gravity in the x and z directions respectively; the rotor hub is in-line with

[*] In some countries, people prefer the yaw and the roll controls switched (type I).

[†] Strictly speaking, R is the rotor span, excluding the rotor hub length. However, since the model helicopter's rotor blade remains nearly rigid due to its short length (0.4 to 0.8 meter), high rotor speed (1300 to 1900 rpm), and hingeless hub design, R is assumed to be the distance between the rotor axis and the rotor tip.

the CG in the y direction. The geometry is sketched in Figure 3.

$$\begin{aligned} v_q^b &= \omega_{IB}^b \times q_b + v_{IB}^b \\ &= R_{IB}^T \dot{R}_{IB} \begin{bmatrix} \ell_r \\ 0 \\ -h_r \end{bmatrix} + R_{IB}^T \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_I \end{aligned} \quad (2)$$

where,

$$R_{IB}^T \dot{R}_{IB} = \begin{bmatrix} \dot{\phi} - \dot{\psi} \sin\theta \\ \dot{\theta}\cos\phi + \dot{\psi}\cos\phi\sin\theta \\ -\dot{\theta}\sin\phi + \dot{\psi}\cos\phi\cos\theta \end{bmatrix} \times \quad (3)$$

The advance ratio, μ, which is the airspeed component parallel to the rotor disk, is the magnitude of the first two elements of v_q^b, and the descent ratio, ν, which is the airspeed component perpendicular to the rotor disk, is the magnitude of the third element of v_q^b.

$$\begin{aligned} \mu &= \frac{1}{R\Omega}\sqrt{(v_{q1}^b)^2 + (v_{q2}^b)^2} \quad (4) \\ \nu &= \frac{1}{R\Omega}|v_{q3}^b| \quad (5) \end{aligned}$$

The rotor solidity, σ, is the ratio between the rotor blade area and the rotor disk area. It indicates how "solid" the rotor disk is,[16] and is taken to be the number of blades (2) times the area of a rotor blade ($2cR$) divided by the area defined by the rotor disk (πR^2).

$$\sigma = \frac{2c}{\pi R}$$

The inflow ratio, λ, is the net value of the descent ratio ν and the induced air velocity, the velocity of the air through the rotor blade, v_i. It is nondimensionalized by $R\Omega$.

$$\lambda = -\nu + \frac{v_i}{R\Omega} \quad (6)$$

The lift curve slope, a, is the slope of the function of the lift vs. angle of attack of the main rotor blade.

Flybar dynamics

The dynamics of the rotor and the flybar are the most significant nonlinearities involved in the creation of the forces and moments on the helicopter. The actuator dynamics include as states the flapping angle and velocity $(\beta, \dot{\beta})$ of the flybar and the position and angular velocity (ξ, Ω) of the main rotor blade. As mentioned before, the flybar plays a major role in augmenting the stability of the helicopter. This system is often called as a Bell-Hiller mixer, because it takes advantage of two different cyclic control systems, as shown in Figure 5. Cyclic control is the mechanism by which the rotor blade's pitch is changed in a rotation so that an unequal distribution of the lift applies a moment around the rotor hub. This moment then provides pitch and roll attitude control as depicted in Figure 2. The Bell-mixer allows the blade pitch to be changed directly from the cyclic servo actuator. It is fast in response, but lacks stability. Meanwhile, the Hiller-mixer allows the pitch of the flybar to be changed instead of the pitch of the blade. The flybar then flaps, and this flapping motion causes the pitch of the main blade to change.

There is a direct relationship between the cyclic input applied to the main blades δ_{cyc} (which is the function of stick commands δ_θ and δ_ϕ) and the cyclic angle of the rotor blades θ_{cyc}. A similar relationship exists between the cyclic input applied to the flybar δ_{fly} and the flapping angle of the flybar β. The orientation of the main blade is given by ξ. Note the 90° phase difference between δ_{cyc} and δ_{fly}, due to the geometry of the rotor/flybar assembly sketched in Figure 5.

$$\begin{aligned} \delta_{cyc}(\xi) &= -\delta_\theta \sin\xi - \delta_\phi \cos\xi \quad (7) \\ \delta_{fly}(\xi) &= \delta_\theta \cos\xi - \delta_\phi \sin\xi \quad (8) \end{aligned}$$

An important assumption at this point is that the rotor system does not apply reaction forces back to the actuators, including the flybar (the flybar is considered to be another actuator to the main blades). This is equivalent to assuming the actuators are able to apply infinite amount of forces to the airfoils. We also neglect the influence of μ and ν on the flybar due to its relatively small wing surface.

When a cyclic input is applied by the pilot, the flybar creates lift which tilts the flybar disk. The flybar acts not only as a main blade angle actuator but also as a stabilizer. If the cyclic input were applied to the main blades only, large control forces on the cyclic servo actuators would be required.[18] By applying the cyclic control to the flybar and allowing the flybar to apply a secondary cyclic input to the main blade, the servo load is significantly reduced.

The flybar is hinged freely on the main axis and rotates around the main axis. Its angle, β, is measured with respect to the plane perpendicular to the main rotor axis. The rotation matrix which relates the flybar position to the body coordinates of the helicopter is denoted R_{BF}; as before, the rotation matrix between the helicopter body frame and the inertial frame is R_{IB}. The rotation matrix relating the flybar frame to the inertial frame is thus the product of the two: $R_{IF} = R_{IB}R_{BF}$.

The pitch angle of the flybar in the helicopter body coordinates is β; its yaw angle is $\xi + 90°$. The rotational inertia of the flybar $\widehat{I}$ is unified as I_f; this is a reasonable assumption because most of the flybar mass is concentrated at the tip region. The angular velocity of the flybar ω_F involves the velocity of the flybar and the main axis simultaneously; it can be found by computing $R_{IF}^T \dot{R}_{IF}$ and extracting terms from the elements in R_{IF}.[15]

$$\begin{aligned} I_f &= \int_{R_1}^{R_2} r^2\, m_F\, dr \\ \widehat{I} &= \operatorname{diag}[\; 0 \quad I_f \quad I_f \;] \\ (\omega_F \times) &= R_{IF}^T \dot{R}_{IF} \end{aligned}$$

The external moment applied on the flybar around the pitch axis is τ_F, which is the aerodynamic lift term. To find the total torque, we integrate along the length of the flybar. Since we will only be interested in the second of these three equations, we will not consider the first and the third elements in any detail.

$$\tau_F = \left[\; \tau_{F_1} \quad \int_{R_1}^{R_2} r\, dL \quad \tau_{F_3} \;\right]^T \qquad (9)$$

The lift element dL depends on the angular velocity of the flybar and its angle of attack θ_{AOA}. The angle of attack of the flybar will be influenced by the pilot input δ_{fly} and the second element of the angular velocity vector of the flybar, denoted ω_{F_2}.

$$\begin{aligned} dL &= \frac{1}{2}\rho(\Omega r)^2 a\, \theta_{AOA}\, c\, dr & (10) \\ \theta_{AOA} &= -\frac{\delta_{fly}}{L_5} - \frac{\omega_{F_2}}{\Omega} & (11) \end{aligned}$$

The roll motion of the flybar only affects the lift created by changing the angle of attack. Because this lift term is already included into τ_F, the roll motion of the flybar is neglected. The yaw motion of the flybar follows the angle of the main rotor blade. Once the external force, angular velocity, and inertia have been defined, the motion of the flybar can be described using the Euler equation.

$$\tau_F = \omega_F \times \widehat{I}\,\omega_F + \widehat{I}\,\dot{\omega}_F \qquad (12)$$

The second of these three equations describes the flapping motion of the flybar. Although the complete expression is quite complex, it can be simplified by considering the pitch and roll motions of the helicopter independently. In addition, the small angle approximation is used for β since it should not exceed ±25°. For example, if roll motion only is considered, the second equation of (12) becomes

$$\ddot{\beta} + \cos\xi\ddot{\phi} - 2\Omega\sin\xi\dot{\phi} - \sin^2\xi\beta\dot{\phi}^2 + \Omega^2\beta =$$
$$\frac{1}{8I_f}\rho\Omega^2 ac(R_2^4 - R_1^4)\left(\frac{\delta_\phi}{L_5}\sin\xi - \frac{\dot{\beta}}{\Omega} - \frac{\dot{\phi}}{\Omega}\cos\xi\right) \quad (13)$$

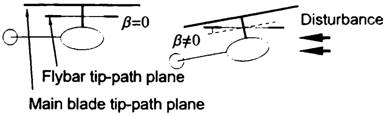


Fig. 4 The stabilizing effect of the flybar. In a hovering situation, the flybar angle β is zero. If a wind gust or other disturbance knocks the helicopter out of its equilibrium, the flybar, which is hinged freely, will continue to rotate in the same inertial plane. Its angle with respect to the main blade becomes nonzero, and it will help bring the helicopter back to equilibrium through its action on the cyclic angle of the main blade.

In the absence of aerodynamic forces and external moments, the flybar behaves as a gyroscope, maintaining its orientation relative to inertial space,[16] as shown in Figure 4. An external disturbance would upset the helicopter angles θ and ϕ, effectively changing β. This nonzero β acts to apply an appropriate compensation input to the main blade cyclic control system to stabilize the helicopter.

Actuator dynamics

Once the flybar dynamics have been found, the rest of the actuator dynamics can be derived. The thrust generated by the rotor blade is T, and the moments represent the moments created by the rotor blade around the roll and the pitch axes are M_ϕ and M_θ respectively.

The moment created by a roll cyclic input is derived by summing the forces around a revolution and along a rotor blade. As per Figure 2, a positive pitch input δ_θ produces roll moment M_ϕ, but acts as pitch moment M_θ due to the precession effect.[16]

$$M_\phi = \frac{1}{2\pi}\int_0^{2\pi}\int_0^R r\sin\xi\, dL_m\, d\xi$$

We assume that the rotor angular velocity is constant; thus $\xi = \Omega t$. The lift element equation is similar to the flybar lift used in equation (10).

$$dL_m = \frac{1}{2}\rho\Omega^2 r^2 a\left(\theta_{cyc} - \frac{v_i}{\Omega r}\right) c dr$$

The induced air velocity v_i must be derived empirically.[16] However, it is usually of similar magnitude around the rotor disk, especially in hover or low-velocity motions. It therefore has little effect on the net moment on the helicopter; we will assume that it is negligible in calculating dL_m.

From Figure 5, there are two different kinds of inputs which affect θ_{cyc}: a direct input by the pilot and an indirect input by the flybar. The geometry of the hub linkages indicate that θ_{cyc} will be the weighted sum of these two inputs.

$$\theta_{cyc} = \frac{L_3}{L_1(L_2+L_3)}\delta_{cyc} + \frac{L_2 L_4}{L_1(L_2+L_3)}\beta \tag{14}$$

The expression for the rotor thrust T is well-known in the literature.[17]

$$T = \frac{1}{2}a\sigma\rho\pi R^4\Omega^2\left(\frac{B^3}{3}\theta_o - \frac{B^2}{2}\lambda + \frac{B^2}{2}\mu\frac{1}{L_1}\sqrt{\delta_\phi^2+\delta_\theta^2}\right) \tag{15}$$

The constant tip loss factor B takes into account the fact that a finite length rotor blade would lose some of the lift generated due to the wing tip vortex effect;[16] we will use the value $B = 0.97$.[17] The throttle stick position δ_t also influences the collective pitch angle θ_o through a simple smooth function, K_{θ_o}. Under normal operating conditions, these two effects balance each other, and Ω is expected to remain constant.[16] Doing so also ensures constant cyclic control gain. Recall that θ_o is the collective pitch angle, and δ_ϕ and δ_θ are the roll and pitch inputs respectively.

(16)

The yaw command δ_ψ directly influences the collective pitch of the tail rotor blades θ_{oT}, and the throttle input δ_t is directly coupled both to the motor torque T_m and the collective pitch of the main rotor blades θ_o. We model these relationships as linear because their dynamics are fast compared to the main rotor dynamics.

$$\begin{aligned}\theta_{oT} &= K_{\theta_{oT}}\cdot\delta_\psi &(17)\\ \theta_o &= K_{\theta_o}\cdot\delta_t &(18)\\ T_m &= K_{T_m}\cdot\delta_t &(19)\end{aligned}$$

The angular velocity of the tail rotor blades is related to the angular velocity of the main rotor blades through a constant K_Ω.

$$\Omega_T = K_\Omega \cdot \Omega$$

The yaw moment M_ψ is equal to the thrust of the tail rotor multiplied by the distance L_T between the main and tail rotor axes. There is no cyclic input for the tail rotor blades; only a collective pitch angle. Thus, the yaw moment equation is similar to the thrust equation of the main rotor blade (15), replacing the terms with the tail rotor equivalents where appropriate.

$$M_\psi = \frac{1}{2}a_T\sigma_T\rho\pi R_T^4\Omega_T^2\left(\frac{B^3}{3}\theta_{oT} - \frac{B^2}{2}\lambda_T\right)L_T \tag{20}$$

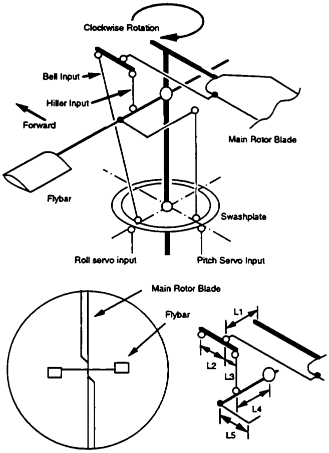


Fig. 5 The basic structure of the model helicopter's cyclic control system. The flapping angle of the flybar, β, is the angle of the flybar with respect to the body coordinate frame attached to the rotor hub. It is zero when the flybar is perpendicular to the rotor axis. Ball joints are shown as "o", and fixed joints are shown as "•". The cyclic pitch input to the main rotor blade is controlled by the combination of the Bell input from the swashplate and the Hiller input from the flybar.

The subscripts T indicate that the values pertain to the tail rotor. To calculate the inflow ratio of the tail rotor λ_T, we can use equation(6). First, equation(2) is used to compute the velocity of the tail rotor hub in body coordinates; q_b must be replaced $[-L_T \;\; 0 \;\; 0]^T$ which is the coordinate of tail rotor hub. Equation (5) can then be used to obtain ν_T. Again, the induced air velocity v_i must be estimated or obtained empirically.

System Identification

The system identification is performed by isolating the effects of each input, and considering only a single output. As a first step, we are attempting to identify some of the physical parameters of the helicopter while restricting to either roll or pitch motion only. The direct least squares method is applied to identify the coefficients of the linearized

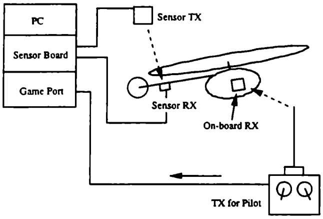


Fig. 6 A sketch of the experimental setup for the system identification of the helicopter's physical parameters. The helicopter is controlled through the radio transmitter by a human pilot. The computer is used to record the input data from the transmitter and the output data from the sensor.

SISO transfer function. The input and output data are collected while a pilot gives either a roll or pitch control input (the other inputs are held to zero).

For example, consider the roll motion of the helicopter. Putting together the roll equation from (1) and the flybar equation (13), it can be seen that the equation relating the roll input δ_ϕ to the roll angle ϕ will be fourth order. The unknown parameters in the equation are the inertias I_{xx} and I_f and the lift curve slope a; the rotor angular velocity Ω can be measured directly. Because the input from the transmitter is held constant over each sample time, a discrete-time transfer function $H_d(z)$ can be used to represent the linearized system:

$$H_d(z) = \frac{b_1 z^4 + b_2 z^3 + b_3 z^2 + b_4 z + b_5}{z^4 + a_1 z^3 + a_2 z^2 + a_3 z + a_4} \qquad (21)$$

The coefficients a_i, b_i will be identified using a least squares method; finding the best match between a linear SISO transfer function and the measured input-output data.

Experimental setup

A Polhemus sensor[3] is used to measure the position and orientation of the helicopter; full six degree-of-freedom information $(x, y, z, \phi, \theta, \psi)$ is available at 50Hz. As shown in Figure 6, the sensor consists of a board connected to the PC's ISA slot, a transmitter, and a receiver. The transmitter is fixed to the ceiling, and it sends out a magnetic field via three orthogonal inductors. The receiver, fixed to the helicopter, senses the strength and the orientation of the magnetic field and sends this information back to the PC. For the preliminary experiments described here, we have fixed the helicopter to a stand to be able to concentrate on the pitch and roll motion. The input and output data are taken at 50 Hz, for a total of approximately 2 minutes duration.

experiment	mean-squared error
pitch SISO	0.0002
roll SISO	0.0010
pitch coupled	0.0013
roll coupled	0.0014

Table 1 A sampling of the mean-squared errors between the actual and simulated outputs using the identified transfer functions $H_{\delta_{\Phi_d}}(z)$ and $H_{\delta_{\theta_d}}(z)$. Although the errors are slightly larger in the coupled experiment, they are still small enough to give some degree of confidence in the identified transfer function.

Results

Once the input-output data has been taken and stored for the isolated roll motion, the system identification algorithm is run to estimate the discrete-time transfer function coefficients a_i and b_i, and thus the SISO transfer function $H_\phi(z)$. The initial condition of the system is also estimated using a least-squares method. The input data and the estimated transfer function are used to simulate the output data, and the simulated and actual output are compared. A reasonable match is achieved, as shown in Figure 7. The mean-squared error between the actual output ϕ_a and the simulated output ϕ_s from the estimated transfer function is computed as follows:

$$error^2 = \frac{1}{N}\sum_{i=1}^{N}(\phi_a(i) - \phi_s(i))^2$$

Typical values for this error on the order of 10^{-3} to 10^{-4}.

A similar identification is performed for the pitch motion, and the SISO transfer function $H_\theta(z)$ is identified. The input data and the estimated transfer function are used to simulate the output data, and the simulated and actual output are compared. Again, a reasonable match is achieved, as shown in Figure 8.

To determine the validity of the single-axis identification, an experiment was performed in which both pitch and roll motions were commanded by the pilot. The input-output data was collected, and the input data was used with the single-input transfer functions to simulate both the pitch and roll outputs. The results from this experiment are given in Figure 9. There is some coupling between the pitch and roll motions, but it does not overwhelm the dominant single-input effects. A sampling of the mean-squared errors is given in Table 1.

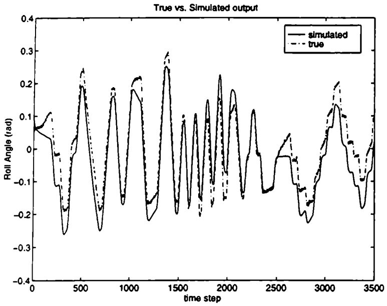


Fig. 7 The comparison between the simulated roll output based on the identified discrete time transfer function and the actual output from the experiment, given the identical roll input command history. The initial value for the simulation was also determined with least squares method with first 50 points from the experiment.

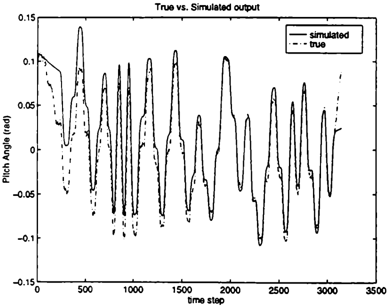


Fig. 8 The comparison between the simulated pitch output based on the identified discrete time transfer function and the actual output from the experiment, given the identical pitch input command history. The initial value for the simulation was also determined with least squares method using the first 50 points from the experiment.

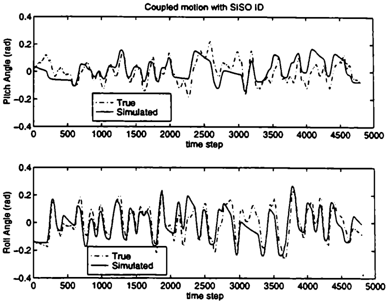


Fig. 9 The system identification result is applied to the pitch and roll motion simultaneously. The figure shows some coupled responses through the discrepancy between the true and the simulated result.

Conclusions and Future Work

A model helicopter is significantly different than a full-scale helicopter—it is faster in response, and lacks stability. A stability augmentation device called a flybar is commonly used to help the pilot control the model helicopter. The main contribution of this paper is to model the interaction between this flybar and the main rotor blade assembly, and to use these dynamic equations to derive a new and complete mathematical model for the dynamics of a model helicopter. Some preliminary system identification experiments were also presented.

We plan to continue the system identification, and use the results to identify the some of the physical parameters of the helicopter such as the inertias, aerodynamic coefficients, and other constants (such as the gains K_{θ_0}, etc.). These parameters will be used to get an accurate nonlinear model for the helicopter dynamics using the equations outlined in the modeling section of this paper. Further experiments will then be performed to validate the nonlinear model. Once we have a fairly accurate nonlinear model, we will begin our feedback control experiments. A short-term goal of this project is to autonomously hover the helicopter in the lab. A longer-term goal is to use the complete nonlinear model to study the interaction between path planning and feedback control for autonomous vehicles, including the model-scale helicopter.

Nomenclature

a	main rotor lift slope
a_T	tail rotor lift slope
B	tip loss factor
c	main rotor blade chord length
$D_{F_{x,y,z}}$	fuselage profile drag forces
dL, dL_m	differential lift elements for flybar and main rotor blade
g	gravitational acceleration
h_r	distance between rotor disk and CG, parallel to rotor axis
$\widehat{I}$	rotational inertia matrix of flybar
I_{3x3}	3x3 identity matrix
I_f	flybar moment of inertia in flapping
I_r	rotor moment of inertia around rotor axis
$I_{xx,yy,zz,xz}$	fuselage rotational moments of inertia
$\mathcal{I}$	rotational inertia matrix of helicopter
K_g	gyro gain for tail rotor
K_{T_m}	proportional constant relating δ_t to T_m
K_{θ_o}	proportional constant relating δ_t to θ_o
$K_{\theta_{oT}}$	proportional constant relating δ_t to θ_{oT}
K_Ω	proportional constant relating Ω to Ω_T
ℓ_r	distance between rotor axis and CG, perpendicular to rotor axis
L_T	distance between tail rotor axis and CG
$L_1 \ldots L_5$	linkage lengths in rotor hub assembly
m	helicopter total mass
m_F	Flybar mass per length
$M_{\phi,\theta,\psi}$	net moments on helicopter
R	length of main rotor blade
R_T	length of tail rotor blade
R_1	distance between rotor axis and flybar tip
R_2	distance between rotor axis and flybar root
$R_{IB,BF,IF}$	rotation matrices between Inertial, Body, and Flybar frames
T	net thrust generated by rotor
T_m	torque applied by motor
v_i	induced air velocity through rotor disk
v_{IB}^b	linear velocity of helicopter
x, y, z	helicopter position coordinates
β	flybar flapping angle
δ_{cyc}	cyclic input displacement
δ_{fly}	cyclic input to flybar
$\delta_\theta, \delta_\phi, \delta_\psi$	roll, pitch, and yaw command input
δ_t	throttle command input
θ_{AOA}	angle of attack of main rotor blade
θ_o	collective pitch angle of main rotor blades
θ_{oT}	collective pitch angle of tail rotor blades
θ_{cyc}	cyclic pitch angle of a main rotor blade
λ	inflow ratio for main rotor
λ_T	inflow ratio for tail rotor
μ	advance ratio
ν	descent ratio
ν_T	descent ratio for tail rotor
ρ	air density
σ	main rotor solidity
σ_T	tail rotor solidity
τ_F	moment applied to flybar
ϕ, θ, ψ	helicopter angular position
ω_{IB}^b	angular velocity of helicopter
ω_F	angular velocity of flybar
Ω	main rotor angular velocity
Ω_T	tail rotor angular velocity

References

[1] Morris, J., Nieuwstadt, M., and Bendotti, P., "Identification and Control of a Model Helicopter in Hover," *Proceedings of the American Control Conference*, Vol. 2, 1994, pp. 1238–1242.

[2] Zhu, X. and Nieuwstadt, M., "The Caltech Helicopter Control Experiment," *CDS Technical Report 96-009*, 1996.

[3] Polhemus Incorporated, P.O. Box 560, Colchester, VT 05446, *3SPACE User's Manual*, Dec. 1993.

[4] Sugeno, M., "Fuzzy hierarchical control of an unmanned helicopter," 1994, available at ftp://ftp.cs.arizona.edu/japan/kahaner.reports/sugeno.94.

[5] Fagg, A., Lewis, M., Montgomery, J., and Bekey, G., "The USC autonomous flying vehicle: An experiment in real-time behavior-based control," *Proceedings of the IEEE Intl Conference on Intelligent Robots and Systems*, Vol. 2, 1993. pp. 422–429.

[6] Montgomery, J., Fagg, A., and Bekey, G., "AFV-I: A behavior-based entry in the 1994 international aerial robotics competition," *IEEE Expert*, Vol. 10, No. 2, Apr. 1994, pp. 16–22.

[7] Johnson, E., DeBitetto, P., Trott, C., and Bosse, M., "The 1996 MIT/Boston Univ/Draper Lab Autonomous Helicopter System," *Proceedings of the 15th AIAA/IEEE Digital Avionics Systems Conference*, 1996, pp. 381–386.

[8] Ozimina, C., Tayman, S., and Chaplin, H., "Flight Control System Design For A Small Unmanned Aircraft," *Proceedings of the American Control Conference*, Vol. 5, Jun. 1995, pp. 2964–2969.

[9] Kaminer, I., Hallberg, E., Pascoal, A., and Silvestre, C., "On the Design and Implementation of a Trajectory Tracking Controller for a Fixed Wing UAV," *Proceedings of the American Control Conference*, Jun. 1995, pp. 2970–2974.

[10] Thornberg, C. and Cycon, J., "Sikorsky Aircraft's UAV, Cypher: System Description and Program Accomplishments," *Proceedings of the American Helicopter Society 51st Annual Forum*, May 1995.

[11] Pallett, T., *Real-time Helicoper Flight Control*. Master's thesis, School of Electrical Engineering, Purdue University, Aug. 1991.

[12] Furuta, K., Ohyama, Y., and Yamano, O., "Dynamics of RC helicopter and Control," *Mathematics and Computers in Simulation XXVI*, 1984, pp. 148–159.

[13] Azuma, A., "Dynamic Analysis of the Rigid Rotor System," *Journal of Aircraft*, Vol. 4, No. 3, 1967.

[14] Padfield, G., *Helicopter Flight Dynamics*, Blackwell Science Ltd, 1996.

[15] Murray, R., Li, Z., and Sastry, S., *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994.

[16] Johnson, W., *Helicopter Theory*, Dover Publications, Inc., 1980.

[17] Okuno, Y., Kawachi, K., Azuma, A., and Saito, S., "Analytical Prediction of Height-Velocity Diagram of a Helicopter Using Optimal Control Theory," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 2, 1991, pp. 453–459.

[18] Hostetler, R., *RAY's complete helicopter manual*, R/C modeler corporation, 1991.

[19] Astrom, K. and Wittenmark, B., *Adaptive Control*, Addison-Wesley Publishing Company, 1995.

AIAA-98-4358

MODELING AND DYNAMIC ANALYSIS OF HELICOPTER UNDERSLUNG SYSTEM[*]

C. Chen, K.Y. Lim, and C.S.P. Seah
DSO National Laboratories
20 Science Park Drive, Singapore 118230

Abstract

This report presents modeling and dynamic analysis of helicopter underslung system. Proper limitations and envelop for the helicopter motion is defined such that the performance of the store is within acceptance. Sensitivity studies are also carried out.

Notations

AoA	angle of attack
$AoSS$	angle of sideslip
c_i	cable i
d	distance between load c.g. and its attachment point
F_x^r, F_y^r, F_z^r	inertial axis components of the total aerodynamic force
g	acceleration due to gravity
$\vec{i}_N, \vec{j}_N, \vec{k}_N$	unit vectors along the inertial axis
$\vec{I}^r$	central inertia dyadic
I_x^r, I_y^r, I_z^r	moments of inertia about (x_r, y_r, z_r) axes
I_{xz}	product of inertia
k, l	cable stiffness constant and length
l_i	instantaneous length of cable c_i
l_0	unstretched length of cable
L_r, M_r, N_r	body axis components of aero. moment
m_r	mass of rigid body r
$\vec{n}_{ab}$	position vector from point a to b
p_r, q_r, r_r	angular velocity components
$\vec{r}_{ab}$	unit vector from a to b
T_i	tension force in cable c_i
u_r, v_r, w_r	translational velocity components
x_r, y_r, z_r	inertial position components
$\dot{x}_r, \dot{y}_r, \dot{z}_r$	inertial velocity components
$\vec{\alpha}^r$	angular acceleration vector
ϕ_r, θ_r, ψ_r	Euler angle triplet
$\dot{\phi}_r, \dot{\theta}_r, \dot{\psi}_r$	Euler angular rate triplet
$\vec{\omega}^r$	angular velocity vector
h	subscript for helicotper
l	subscript for load

1. Introduction

The capability of using helicopter to transport underslung store has been addressed in the civil and military rotorcraft arenas since the 1930s[1]. This allows for the transport of bulky or oversized loads and cargo to the location where it is not practical to land the helicopter.

Underslung store presents additional dynamic issues to the already complex dynamic system of the helicopter. For example, the forward speed of a helicopter can be severely restricted by the onset of dynamic instability of the load[2]. An interesting review of the dynamic problems of the helicopter underslung system can be found in Sheldon's paper[3].

In this paper, the underslung configuration will be a helicopter hanging the 2.0 × 2.0 × 2.0 m cargo store with 4 elastic cables (see Figure 1). Equations of motion of the system will be built-up. A full range of aerodynamic forces and moments for the store will also be worked out.

Moreover, the interested store in this paper is a very light store. In reality, a light store brings more dynamic problem than a heavier store. Its behavior becomes unpredictable even when a helicopter flights at not-so-high forward velocity.

It is not surprised to find that, in the real flight test, sometimes the store is very ***stable***: moving with helicopter, keeping certain attitude; sometimes the store can be very ***unpredictable***: sudden load movement causing the cables to slacken or collapse. Therefore, a question is naturally asked: what kind of states of the store is acceptable? As such, classification of the store behavior will be defined in the paper.

As concluded, the objective of this paper is to perform first cut analysis on the dynamics of helicopter underslung system to determine the store behavior and recommend a preliminary helicopter flight envelope for safe underslung operations, particular when weight of an underslung store is very small.

2 Aerodynamics Modeling

The flow around the single cargo store (2.0 x 2.0 x 2.0m) was studied using Navier-Stokes Class CFD tool (StarCD) with k-e/RNG Turbulence Modeling. The purpose was to obtain full range aerodynamics coefficient of the store (Angle of Attack AoA -180^o to 180^o, Angle of Side Slip AoSS -180^o to 180^o), for the generation of the store aerodynamic forces and moments.

2.1 Numerical Analysis Set-up

The computational mesh was constructed using a total of 117,185 cell density, seen from Table 1. The mesh density was generally higher at regions where recirculatory flow was expected. Also, in order to predict the boundary profiles adjacent to the wall and the viscosity effects accurately, the grids were generally very fine near to the wall. This is very important especially during high flow separation or high AoA.

Since our store is a non-streamline body, it is also classified as a bluff body. Generally, for bluff bodies, majority of the forces are contributed due to pressure forces arising from separation of boundary layers flow adjacent to the surface over the rearward facing part of the body. Hence CFD grid density was generally higher in those regions

2.2 Assumptions

From a few sample CFD runs, it was observed that the store aerodynamic coefficients were relatively insensitive to the airspeed. Hence all the runs were carried out at 30m/s.

This assumption is valid since:

1. though the forces and moments were different for different speeds, their respective non-dimensional coefficients are normalized by the dynamics pressure and its area. $q = \frac{1}{2}\rho v_\infty^2$
2. within the operational speed of 30 to 120 knots, the flow remains in the incompressible regions.
3. the Reynolds number remains fairly constant within the specified operating range. (from 1×10^6 to 4×10^6)

Also, using the symmetrical properties of the store, we reduced the runs required from 36x36 combinations to 18x18 combinations ($AoA - 180^o$ to 180^o, $AoSS - 180^o$ to 180^o) to obtain the full range aerodynamics, thus reducing the number of CFD runs needed. The rest of the data were reflected using symmetry rules to obtain the full aerodynamic coefficient of the store.

2.3 Results

The full range (AoA -180^o to 180^o, AoSS -180^o to 180^o) aerodynamic coefficients of the store (consisting of C_x, C_y, C_z, C_{m_x}, C_{m_y} and C_{m_z}) is used as a database for input into the in-house dynamics Model for simulating the store aerodynamics.

With the provided aerodynamics data, the dynamics model will be able to simulate the aerodynamic forces and moments on the store during flights.

Yaw Moment

Since the store analyzed was modeled as a cube shaped store, for $AoSS < 45^o$ and $AoSS > 45^o$, a restoring moment was expected to yaw the store towards 45^o due to difference in pressure distribution (refer to Figure 2). Note that at $AoSS = 45^o$, due to symmetry of the store, a symmetrical pressure distribution was expected on both side of the face, which ultimately yields zero yawing moment.

From the plot of the predicted Yaw Moment Coefficient *v.s.* AoSS (refer to Figure 3), it was confirmed that the yaw was approximately zero at $AoSS = 45^o$ for various AoA, hence partially verifying the CFD results through our observation. Also, another implication of this observation was that during the steady flight condition in the dynamics analysis, we would expect the store to oscillate approximately about the zero yaw angle ($AoSS = 45^o$ for a cubic store).

Verification of Results

To verify the results, the C_x and C_y (body axis) obtained was compared against the ESDU[4] (Engineering Sciences Data Unit) results. From the plot (see Figures 4), it could be seen that StarCD managed to predict the C_x reasonably accurately, well within the 20% error expected from the ESDU under non-datum conditions. Most importantly, StarCD captured most of the trends rather closely.

The quality of the near wall mesh was also checked by obtaining the value of the y^+. y^+ is a dimensionless normal distance from the walls.

$$y^+ = \frac{V_\tau y}{v} \tag{1}$$

Our CFD runs show that $y^+ \approx 30$ for majority of the store surface. This means that the grid is close enough near the surface of the cube, such that the "near-wall function" capability of the turbulence model is able to predict correctly the boundary layer profiles that form adjacent to the walls. In short, the quality of grids next to the wall is good enough to simulate closely the actual flow around the store.

3. Nonlinear Dynamic Model

The equations of motion (EOM) of helicopter underslung system would be built up in this section[5]. Since the underslung store was connected with helicopter using 4 elastic cables, the system itself was quite complicated. The methodology developed below can be used not only in the current case, but also in other complicated underslung systems, *e.g.*, two-hooks, spreader bar, and multi-lift system[6].

3.1 Axis Systems

Three kinds of reference frames would be adopted in describing the EOM of underslung load systems, seen from Figure 5. There were:

Inertial Axis System: an $\vec{i}_N$, $\vec{j}_N$, $\vec{k}_N$ coordinate system F_N fixed to the surface of the Earth in such a manner that $\vec{k}_N$ pointed downward, $\vec{i}_N$ pointed to the right, and $\vec{j}_N$ such that a right-handed coordinate system was formed.

Helicopter Body Axis System: an $\vec{i}_h$, $\vec{j}_h$, $\vec{k}_h$ coordinate system F_h fixed to c.g. of the helicopter in such a manner that $\vec{i}_h$ pointed along the longitudinal axis of symmetry of the helicopter, $\vec{k}_h$ points towards the surface of the Earth, and $\vec{j}_h$ such that a right-handed coordinate system was formed.

Load Body Axis System: an $\vec{i}_l$, $\vec{j}_l$, $\vec{k}_l$ coordinate system F_l fixed to c.g. of the load in such a manner that $\vec{i}_l$ pointed along the longitudinal axis of symmetry of the load, $\vec{k}_l$ pointed towards the surface of the Earth, and $\vec{j}_l$ such that a right-handed coordinate system was formed.

It was well-known that the transformation matrices associated with single rotations about the three coordinate axes were as follows:

$$E_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \tag{2}$$

$$E_2(\theta) = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \tag{3}$$

$$E_3(\psi) = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The transformation relationship between the inertial axis system F_N and any rigid body axis system F_r was given by:

$$T^{N,r}(\psi_r, \theta_r, \phi_r) = E_3(\psi_r)E_2(\theta_r)E_1(\phi_r) \tag{5}$$

where $T^{N,r}$ defined transformation matrix from rigid body r to inertial frame N.

3.2 Force and Moment Equations

The EOM would be obtained by first considering vectorial force and moment balance equations of two rigid bodies. The vectorial equations would then be resolved along appropriate coordinate axes.

Vectorial Form

Firstly, considered the force balance of the helicopter:

$$\begin{aligned} &(F_x^h - m_h\ddot{x}_h)\vec{i}_N + (F_y^h - m_h\ddot{y}_h)\vec{j}_N + \\ &(F_z^h + m_h g - m_h\ddot{z}_h)\vec{k}_N + T_{12}\vec{r}_{12} + \\ &T_{13}\vec{r}_{13} + T_{14}\vec{r}_{14} + T_{15}\vec{r}_{15} = 0 \end{aligned} \tag{6}$$

where T_{12}, T_{13}, T_{14} and T_{15} define tension forces along the cables c_{12}, c_{13}, c_{14} and c_{15}, respectively. The moment balance of the helicopter about the attachment point can be written as follows:

$$\begin{aligned} &\vec{n}_{11\bullet} \times [(F_x^h - m_h\ddot{x}_h)\vec{i}_N + (F_y^h - m_h\ddot{y}_h)\vec{j}_N + \\ &(F_z^h + m_h g - m_h\ddot{z}_h)\vec{k}_N] + L_h\vec{i}_h + M_h\vec{j}_h + \\ &N_h\vec{k}_h - \vec{\alpha}^h \cdot \vec{I}^h - \vec{\omega}^h \times \vec{I}^h \cdot \vec{\omega}^h = 0 \end{aligned} \tag{7}$$

Next, considering the force balance of the underslung load resulted in:

$$\begin{aligned} &(F_x^l - m_l\ddot{x}_l)\vec{i}_N + (F_y^l - m_l\ddot{y}_l)\vec{j}_N + \\ &(F_z^l + m_l g - m_l\ddot{z}_l)\vec{k}_N - T_{12}\vec{r}_{12} - \\ &T_{13}\vec{r}_{13} - T_{14}\vec{r}_{14} - T_{15}\vec{r}_{15} = 0 \end{aligned} \tag{8}$$

Since underslung load system only had one attachment point, there would not be enough equations to solve a number of unknown variables. From the dynamics point of view, we can treat point 1 as two different points, *i.e.*, 1 and 1', as shown in Figure 6. From this sense, the moment balance equation of the load about the imaginary attachment point 1' was given as follows:

$$\vec{r}_{1'2\bullet} \times [(F^l_x - m_l\ddot{x}_l)\vec{i}_N + (F^l_y - m_l\ddot{y}_l)\vec{j}_N + (F^l_z + m_lg - m_l\ddot{z}_l)\vec{k}_N] + L_l\vec{i}_l + M_l\vec{j}_l + N_l\vec{k}_l - \vec{\alpha}^l \cdot \vec{I}^l - \vec{\omega}^l \times \vec{I}^l \cdot \vec{\omega}^l = 0 \tag{9}$$

Basically, there were four sources of forces acting on the load: (a) gravity force, (b) inertial force, (c) cable tension force, and (d) aerodynamic force. The gravity force were represented by the weight acting in the load c.g. location, while the inertial force were described by the components of linear motion of the c.g., and by the components of the rotational motion about the c.g. Cable tension force were simply obtained using spring stiffness property. Aerodynamic force were pressures distributed over the load surface and were customarily described by the following six quantities:

$$\begin{aligned} X_l &= \tfrac{1}{2}\rho V_l^2 SC_x(\alpha,\beta) \\ Y_l &= \tfrac{1}{2}\rho V_l^2 SC_y(\alpha,\beta) \\ Z_l &= \tfrac{1}{2}\rho V_l^2 SC_z(\alpha,\beta) \\ L_l &= \tfrac{1}{2}\rho V_l^2 Sl_bC_{m_x}(\alpha,\beta) \\ M_l &= \tfrac{1}{2}\rho V_l^2 Sl_aC_{m_y}(\alpha,\beta) \\ N_l &= \tfrac{1}{2}\rho V_l^2 Sl_bC_{m_z}(\alpha,\beta) \end{aligned} \tag{10}$$

where ρ was the air density, V_l was the free stream velocity, S, l_a and l_b were the reference area, geometric chord, geometric span of the load, respectively. C_x, C_y, C_z, C_{m_x}, C_{m_y} and C_{m_z} were six aerodynamic force and moment coefficients, discussed in Section 2. It was noted that X_l, Y_l, Z_l were obtained along body axis of the load. By using transformation matrix $T^{N,l}$ from load axis system to inertial space resulted in:

$$\begin{bmatrix} F^l_x \\ F^l_y \\ F^l_z \end{bmatrix} = T^{N,l}(\psi_l,\theta_l,\phi_l)\begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix} \tag{11}$$

Scalar Form

Considering the scalar components of Equation (6), the difficulty was to derive cable orientation, which were $\vec{r}_{12}$, $\vec{r}_{13}$, $\vec{r}_{14}$, and $\vec{r}_{15}$. The procedure of solving $\vec{r}_{12}$ would be illustrated here and the solutions for the rest three vectors would be directly given. One kinematic equation can be employed from Figure 6:

$$\vec{n}_{12} = \vec{n}_{1\bullet 2\bullet} - \vec{n}_{1\bullet 1} - \vec{n}_{22\bullet} \tag{12}$$

where $\vec{n}_{1\bullet 1} = h\vec{k}_h$. Here, h was the distance between the helicopter c.g. location and its cable securing point 1.

Assuming that length, width and height of load were $2a$, $2b$ and $2c$, respectively, the scalar components of vector $\vec{n}_{22\bullet}$ can be decided by using coordinate transformation

$$\begin{bmatrix} x_{B_{22\bullet}} \\ y_{B_{22\bullet}} \\ z_{B_{22\bullet}} \end{bmatrix} = T^{N,r}(\psi_r,\theta_r,\phi_r)\begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{13}$$

The scalar components of vector $\vec{n}_{1\bullet 1}$ can be given as:

$$\begin{aligned} x_{B_{1\bullet 1}} &= h(cos\psi_h sin\theta_h cos\phi_h + sin\psi_h sin\phi_h) \\ y_{B_{1\bullet 1}} &= h(sin\psi_h sin\theta_h cos\phi_h - cos\psi_h sin\psi_h) \\ z_{B_{1\bullet 1}} &= hcos\theta_h cos\phi_h \end{aligned} \tag{14}$$

By using Equations (12), (13) and (14), the scalar components of vector $\vec{n}_{12}$ can be obtained:

$$\begin{aligned} x_{B_{12}} &= x_l - x_h - x_{B_{1\bullet 1}} - x_{B_{22\bullet}} \\ y_{B_{12}} &= y_l - y_h - y_{B_{1\bullet 1}} - y_{B_{22\bullet}} \\ z_{B_{12}} &= z_l - z_h - z_{B_{1\bullet 1}} - z_{B_{22\bullet}} \end{aligned} \tag{15}$$

The cable c_{12} actual length l_{12} should satisfy

$$l_{12}^2 = x_{B_{12}}^2 + y_{B_{12}}^2 + z_{B_{12}}^2 \tag{16}$$

and the unit vector $\vec{r}_{12}$, standing for orientation of the cable c_{12}, was

$$\vec{r}_{12} = \frac{x_{B_{12}}}{l_{12}}\vec{i}_N + \frac{y_{B_{12}}}{l_{12}}\vec{j}_N + \frac{z_{B_{12}}}{l_{12}}\vec{k}_N \tag{17}$$

Using the same methodology, orientation of other three cables can be obtained. As such, the desired form of the scalar components of Equation (6) was:

$$F^h_x - m_h\ddot{x}_h + T_{12}\frac{x_{B_{12}}}{l_{12}} + T_{13}\frac{x_{B_{13}}}{l_{13}} + T_{14}\frac{x_{B_{14}}}{l_{14}} + T_{15}\frac{x_{B_{15}}}{l_{15}} = 0 \tag{18}$$

$$F^h_y - m_h\ddot{y}_h + T_{12}\frac{y_{B_{12}}}{l_{12}} + T_{13}\frac{y_{B_{13}}}{l_{13}} + T_{14}\frac{y_{B_{14}}}{l_{14}} + T_{15}\frac{y_{B_{15}}}{l_{15}} = 0 \tag{19}$$

$$F^h_z + m_hg - m_h\ddot{z}_h + T_{12}\frac{z_{B_{12}}}{l_{12}} + T_{13}\frac{z_{B_{13}}}{l_{13}} + T_{14}\frac{z_{B_{14}}}{l_{14}} + T_{15}\frac{z_{B_{15}}}{l_{15}} = 0 \tag{20}$$

The scalar components of Equation (8) were derived in a manner similar to that used for Equation (6). These equations were:

$$F^l_x - m_l\ddot{x}_l - T_{12}\frac{x_{B_{12}}}{l_{12}} - T_{13}\frac{x_{B_{13}}}{l_{13}} - T_{14}\frac{x_{B_{14}}}{l_{14}} - T_{15}\frac{x_{B_{15}}}{l_{15}} = 0 \tag{21}$$

$$F^l_y - m_l\ddot{y}_l - T_{12}\frac{y_{B_{12}}}{l_{12}} - T_{13}\frac{y_{B_{13}}}{l_{13}} - T_{14}\frac{y_{B_{14}}}{l_{14}} - T_{15}\frac{y_{B_{15}}}{l_{15}} = 0 \tag{22}$$

$$\begin{aligned}&F_z^l + m_l g - m_l \ddot{z}_l - T_{12}\frac{z_{B_{12}}}{l_{12}} - T_{13}\frac{z_{B_{13}}}{l_{13}} - \\ &T_{14}\frac{z_{B_{14}}}{l_{14}} - T_{15}\frac{z_{B_{15}}}{l_{15}} = 0\end{aligned} \tag{23}$$

The scalar components of Equation (7) along the body axes of the helicopter were given by:

$$\begin{aligned}&\dot{p}_h I_x^h + \dot{r}_h I_{xz}^h - q_h r_h (I_y^h - I_z^h) + p_h q_h I_{xz}^h + \\ &h_1[(F_x^h - m_h \ddot{x}_h)(sin\psi_h cos\phi_h - \\ &cos\psi_h sin\theta_h sin\phi_h) - (F_y^h - m_h \ddot{y}_h) * \\ &(sin\psi_h sin\theta_h cos\phi_h + cos\psi_h cos\phi_h) - \\ &(F_z^h + m_h g - m_h \ddot{z}_h) cos\theta_h sin\phi_h] - L_h = 0\end{aligned} \tag{24}$$

$$\begin{aligned}&\dot{q}_h I_y^h - p_h r_h (I_z^h - I_x^h) + (r_h^2 - p_h^2) I_{xz}^h + \\ &h_1[(F_x^h - m_h \ddot{x}_h) cos\psi_h cos\theta_h + \\ &(F_y^h - m_h \ddot{y}_h) sin\psi_h cos\theta_h - \\ &(F_z^h + m_h g - m_h \ddot{z}_h) sin\theta_h] - M_h = 0\end{aligned} \tag{25}$$

$$\begin{aligned}&\dot{r}_h I_z^h + \dot{p}_h I_{xz}^h - p_h q_h (I_x^h - I_y^h) - \\ &q_h r_h I_{xz}^h - N_h = 0\end{aligned} \tag{26}$$

where p_h, q_h, and r_h were the body axis components of the helicopter angular velocity, $\vec{\omega}^h$.

Scalar components of vector $\vec{n}_{1'2*}$ from load moment equation (9) can be obtained in a similar manner as $\vec{n}_{12}$. Thus, the load moment equation (9) when resolved along the load body axes yielded

$$\begin{aligned}&\dot{p}_l I_x^l - q_l r_l (I_y^l - I_z^l) - y_{B_{1'2*}}^l [(F_x^l - m_l \ddot{x}_l) * \\ &(cos\psi_l sin\theta_l cos\phi_l + sin\psi_l sin\phi_l) \\ &+(F_y^l - m_l \ddot{y}_l)(sin\psi_l sin\theta_l cos\phi_l - cos\psi_l sin\phi_l) \\ &+(F_z^l + m_l g - m_l \ddot{z}_l) cos\theta_l cos\phi_l] \\ &+z_{B_{1'2*}}^l [(F_x^l - m_l \ddot{x}_l) * \\ &(cos\psi_l sin\theta_l sin\phi_l - sin\psi_l cos\phi_l) \\ &+(F_y^l - m_l \ddot{y}_l)(sin\psi_l sin\theta_l sin\phi_l + cos\psi_l cos\phi_l) \\ &+(F_z^l + m_l g - m_l \ddot{z}_l) cos\theta_l sin\phi_l)] - L_l = 0\end{aligned} \tag{27}$$

$$\begin{aligned}&\dot{q}_l I_y^l - p_l r_l (I_z^l - I_x^l) + x_{B_{1'2*}}^l [(F_x^l - m_l \ddot{x}_l) * \\ &(cos\psi_l sin\theta_l cos\phi_l + sin\psi_l sin\phi_l) \\ &+(F_y^l - m_l \ddot{y}_l)(sin\psi_l sin\theta_l cos\phi_l - cos\psi_l sin\phi_l) \\ &+(F_z^l + m_l g - m_l \ddot{z}_l) cos\theta_l cos\phi_l] \\ &-z_{B_{1'2*}}^l [(F_x^l - m_l \ddot{x}_l) cos\psi_l cos\theta_l \\ &+(F_y^l - m_l \ddot{y}_l) sin\psi_l cos\theta_l \\ &-(F_z^l + m_l g - m_l \ddot{z}_l) sin\theta_l] - M_l = 0\end{aligned} \tag{28}$$

$$\begin{aligned}&\dot{r}_l I_z^l - p_l q_l (I_x^l - I_y^l) - x_{B_{1'2*}}^l [(F_x^l - m_l \ddot{x}_l) * \\ &(cos\psi_l sin\theta_l sin\phi_l - sin\psi_l cos\phi_l) \\ &+(F_y^l - m_l \ddot{y}_l)(sin\psi_l sin\theta_l sin\phi_l + cos\psi_l cos\phi_l) \\ &+(F_z^l + m_l g - m_l \ddot{z}_l) cos\theta_l sin\phi_l)] \\ &+y_{B_{1'2*}}^l [(F_x^l - m_l \ddot{x}_l) cos\psi_l cos\theta_l \\ &+(F_y^l - m_l \ddot{y}_l) sin\psi_l cos\theta_l \\ &-(F_z^l + m_l g - m_l \ddot{z}_l) sin\theta_l] - N_l = 0\end{aligned} \tag{29}$$

where p_l, q_l and r_l were the body axis components of the load angular velocity $\vec{\omega}^l$.

4. Nonlinear Simulation

After equations of motion for the helicopter underslung system were built-up, nonlinear simulation using Matlab was then carried out to study the store behavior. In the current stage, only the store behavior was analyzed. The helicopter was assumed to be a point mass towing the store. This was a practical and reasonable assumption because of the following factors:

1. The interested store weight was from 200 *kg* to 700 *kg*. This was insignificant compared with the weight of the helicopter (from 4450 *kg* to 9000 *kg*). Therefore, it was predicted that the store motion would not have a significant effect to the helicopter.
2. If the helicopter was in the steady flight state, the helicopter motion would not have a significant effect on the load motion.
3. The results from store motion can also be used as reference solutions if entire helicopter/store system was studied

Nevertheless, it is still important to study the helicopter underslung system completely and thoroughly in order to have a clearer and more reliable systematic prediction. The work will be continued in a separate project.

Some important parameters used in the simulation were listed out in Table 2. Other initial conditions include $x_l = y_l = 0\ m$, $\dot{x}_l = \dot{y}_l = \dot{z}_l = 0\ m/s$, $\phi_l = \theta_l = 0\ deg$, $\psi_l = -0.001\ deg$, $p_l = q_l = r_l = 0\ deg/s$. Store weight was set as 200 *kg*. This weight would be varied in the sensitivity study in Section 7.

5 Classification of Store Behavior

The basic idea of classification of the store behavior was to answer a question: what kind of states of the store was acceptable? Sometimes the store was very "stable": moving with helicopter, keeping certain attitude; sometimes the store can be very "unpredictable": sudden load movement causing the cables to slacken or collapse. Therefore, it was important and necessary to define different levels of the store behavior such that the pilot would know clearly the limits of helicopter operations with underslung store.

On the basis of the simulation results, three levels of the store behavior were defined as follows:

Ideal: The store was such that

1. the translational motion of the store would follow that of the helicopter
2. at most 1 cable would slacken at one time
3. the store would oscillate mildly about a certain angle (for example, $\pm 45^o$)

Acceptable: The store was such that

1. the translational motion of the store would follow that of the helicopter
2. 2 cables would slacken for a short period
3. the store would continue turning in one direction

Failed: The store was such that

1. more than 2 cables would slacken or even collapse
2. there was sudden change in the store's attitude

Initially, Condition 2 in the *Ideal* behavior was set that no cable would slacken. After discussion with test pilot, we realized that this condition was too conservative. Actually, in reality, the slack of 1 cable in the 4-cable underslung configuration could be frequently observed. Pilot would still classified it as ***ideal*** behavior since it wouldn't have bad effect to the overall operations.

As concluded, in these 3 levels, the first level was the most preferable. Although in the second level, the store showed moderate deficiencies, its state was still acceptable. The third level showed major deficiencies since not only the store but also the helicopter would be badly affected.

6 Motion Study

In this section, behavior of the store would be predicted through simulation in order to recommend a candidate envelope for the desired underslung operation. The interested helicopter motions were: (1) acceleration; (2) cruise; (3) deceleration; (4) climb; (5) descent.

6.1 Acceleration

In the acceleration stage, the store would swing backwards. The relative air mass would hit the upper surface of the store and thus produced a negative angle-of-attack(α). This negative angle induced a positive Z_l force which pulled the store downwards.

Referring to the relationship between yaw moment coefficient(C_{m_z}) and sideslip angle(β), it can be found that $\beta = 0, \pm 90^o, \pm 180^o$ were statically unstable points, while $\beta = \pm 45^o, \pm 135^o$ were statically stable points. This meant that the store had the tendency to place itself diagonally at those statically stable points.

Acceleration Envelope

If there was no aerodynamic effect, the store would swing from vertical centerline to the left and swing back towards the centerline. The longitudinal pendulum angle would oscillate in the range of $[-2\ tan^{-1}a/g\quad 0]$ in the longitudinal plane when the helicopter was doing the forward acceleration motion at $a\ m/s^2$. This phenomenon can be observed in the case of weak aerodynamic effect occurring, as in the first few seconds when the helicopter starts moving. However, as the local velocity of the store increased, the aerodynamic effect became more and more significant. This would cause the store to swing further backwards. For a given helicopter, the mechanical hooking assembly beneath the fuselage would only allow the store oscillate to a cone with a 60^o top angle, seen from Figure 7. This meant the line between the hook and store c.g. location can not exceed 30^o from the vertical centerline. This limitation gave maximum acceleration period of the helicopter, as shown in Figure 8. For example, if the helicopter accelerated at 1.0 m/s^2, the maximum period for the pilot to continue the acceleration motion was 20.0 s. The period was also known as acceleration envelop.

Frequency Domain Analysis

Since the store was connected with 4 elastic cables, it can not be treated as a simple pendulum. A practical way of deriving motion period was to use Fast Fourier Transformation(FFT). In frequency domain analysis, we considered time history response $\dot{z}_l$ as our FFT input signal. It would be sampled at 20 Hz and a Matlab-based 1024-point FFT algorithm was used. The selection of signal $\dot{z}_l$ was that not only store pendulum motion period but also cable vertical bounce motion period can be easily found.

From the FFT results, two dominant frequencies were observed. The frequency of store pendulum motion was equal to 0.1758 Hz, or 5.6889 s in period. The frequency of cable vertical bounce motion was 2.8500 Hz, or 0.35 s in period.

Period for a simple pendulum was calculated using the following well-known formula:

$$T_p = 2\pi\sqrt{\frac{l}{g}} = 3.9\ s \qquad (30)$$

It was noted that this theoretical result was smaller than 5.6889 s when the store mass was 200 kg and cable length was 3.3 m. Generally, for a complicated pendulum, the pendulum motion period was mainly affected by cable length, store and aircraft weight, and underslung configuration. Its value would normally be larger than the period of a simple pendulum motion[7].

For a given helicopter, it was noted that the dutch roll frequency stated in the technical specification was between $0.2 - 0.33\ Hz$. This range was just slightly above store pendulum frequency. Therefore, it was possible that resonance in the coupled store-helicopter system may happen. To avoid possible resonance, the pilot may initiate slight "G" to the store by slight roll or pull up.

The vertical bounce frequency of the coupled aircraft and underslung store can be calculated using the following formula, also from [7]:

$$f_v = \frac{1}{2\pi}\sqrt{\frac{K(M_h + M_l)}{M_h M_l}} \qquad (31)$$

where K was the effective spring stiffness of suspension, which included effect of multiple slings and apex angles. Using this formula, the theoretical value for vertical bounce frequency was 2.9093 Hz when store mass was 200 kg. This value was quite close to 2.8500 Hz from FFT analysis, with relative error being 2.04%. The comparison here also validated the correctness of the in-house model.

It was also suggested from [7] that vertical bounce frequency should not greater than a threshold frequency f_{th}, which was calculated as 0.6 of the aircraft first harmonic frequency. For the given helicopter, its first harmonic frequency is 4.4167 Hz. Therefore, f_{th} is equal to 2.65 Hz. It is noted that vertical bounce frequency was 2.8500 Hz when store mass was 200 kg, which was already beyond f_{th}. Thus, there existed 2 options: 1) to increase store mass; 2) to increase cable length. These 2 options were both served the purpose of decreasing the so-obtained vertical bounce frequency. From calculations, it was suggested that the minimal store mass was set as 240 kg while cable length kept at 3.3 m, or the minimal cable length was set as 4.1 m while store mass kept at 200 kg.

6.2 Cruise

Using the classification of store behavior, the maximum acceptable speed of the helicopter can be found. When $\dot{x}_h <= 20\ m/s$, the behavior of the store was ***ideal***. When $\dot{x}_h = 22.5\ m/s(43.2\ knots)$, it became ***failed***. Actually, it was found that there was very small difference of maximum velocity between ***ideal*** and ***failed***. Thus, the maximum acceptable velocity we recommend was 20 $m/s((38.85\ knots)$.

6.3 Deceleration

Different from the acceleration motion, aerodynamic drag acting on the store prevents it from going up. Unless the deceleration rate is very high, the store has little chance to hit 30° cone angle. Therefore, there exist a maximum deceleration rate: $-1.75\ m/s^2$ when the store weight is 200 kg.

6.4 Climb

The numerical results from Figure 9 suggested that at different forward speed, the helicopter should have different climb rate. When the helicopter was at hovering state, it can climb at 4330 ft/min. When the helicopter flew at 38 $knots$, it can only climb at 394 ft/min. This stated that the faster the forward speed, the smaller the climb rate.

6.5 Descent

The numerical results from Figure 10 recommended that, in order to be have ***ideal*** store behavior, at different forward speed, the helicopter should have different descent rate. When the helicopter flew from hovering state to 19 $knots$, it can descent at 1772 ft/min. The descent rate dropped to 394 ft/min when the helicopter flew from 30 $knots$ to 40 $knots$. This also stated that the faster the forward speed, the smaller the descent rate.

7 Sensitivity Study

Sensitivity study would concentrate on 3 basic physical variables: (1) store mass; (2) cable length; (3) cable spring constant.

Methodology of sensitivity study was as follows: for each variable, there was a nominal value. A set of data points perturbed from nominal value of each variable would be chosen. Nonlinear simulation would then be run based on these data points. Conclusions would finally be drawn on the basis of simulation results.

7.1 Store Mass

Store mass had very significant effects on the dynamic behavior of helicopter underslung system. In Section 6, all the simulation results were based on a setting that mass of load was 200 kg. It was required than the mass would change from 200 kg to 700 kg. Therefore, besides $m = 200\ kg$, store mass was also chosen at 450 kg and 700 kg. Results have been shown in Figure 8. From the figure, it clearly showed that with the store mass increased, acceleration envelope also increased. This was because it took more drag, which led to longer period, to overcome heavier weight.

As to the maximum speed for the helicopter, it can reach 38 m/s (73.866 knots) when the mass of load was 450 kg. It can also reach 48 m/s (93.305 knots) when the mass of load was 700 kg.

For the deceleration, it is found that when the store weight is between 200 kg and 450 kg, the maximum deceleration rate is $-1.75\ m/s^2$. When the store weight is between 450 kg to 700 kg, the rate is $-1.50\ m/s^2$. For the climb and descent, their corresponding envelope can be referred to Figures 9 and 10.

As to the frequency analysis, it is found that changed mass had very little effect to the pendulum motion frequency. In the meantime, vertical bounce frequency was affected significantly as store mass changed, as shown in Table 3. From the table, it showed that our model can well-predict vertical bounce frequency with relative error from 0.27% to 2.04% as compared with theoretical formula. On the contrary, we can also use theoretical formula to easily calculate system vertical bounce frequency. To be compared, it also listed the vertical bounce frequency results when the mass of helicopter was considered.

7.2 Cable Length

Cable length mainly affected the period of load pendulum motion. As stated in Section 5, the load with 4 elastic cables was a complex pendulum system. To validate our results and also to do sensitivity study, different lengths of the cables were discussed here. Besides $l_0 = 3.3\ m$, the natural length of cable can be 5.0 m, 10.0 m, and 33 m. A simplified load model was built using NASTRAN, a structural dynamics software, to calculate different periods for different cable lengths. Comparisons were made between in-house model and NASTRAN model, as shown in Table 4. Seen from the table, it showed that results from two different sources, *i.e.*, In-house and NASTRAN, were very well consistent.

7.3 Cable Spring Constant

As provided, cable spring constant used in the previous simulation was 18182 N/m. Practically, the cables used in the real application, although made from the same material, would not have exactly the same structural property. Two cases would be simulated when the spring constant perturbed $\pm 10\%$ from its nominal value, *i.e.*, 16364 N/m and 20000 N/m. The mission specified was the bank-to-turn motion, as discussed in Section 6. From the simulation, it unveiled a truth that the cable spring constant did not play an important roll in the helicopter underslung system.

8 Conclusions

In this paper, the system for a helicopter hanging the 2.0 × 2.0 × 2.0 m cargo store with 4 elastic cables has been studied in the sense of both aerodynamics and dynamics. Store behavior has been determined and preliminary helicopter flight envelope for safe underslung operations has been recommended.

Acknowledgment

The authors would like to thank Dr. Ho Quan Wai, Dr. Tan Jiak Kwang, and Chew Aik Bin for their invaluable guidance and discussion throughout this project.

References

[1] H. Glauert, *The stability of a body towed by a light wire*, ARC. R and M 1312, 1930

[2] A. Prabhakar, *Stability of a helicopter carrying an underslung load*, Vertica, Vol. 2, 1978

[3] D.F. Sheldon, *An appreciation of the dynamic problems associated with the external transportation of loads from a helicopter – state of the art*, Vertica, Vol. 1, 1977

[4] –, *ESDU Data Items 71016*, Engineering Science Data Unit International, Mclean, Va.

[5] C. Chen, *Equations of motion of underslung load systems with elastic cables*, DSO Report GC/00493/97, 1997

[6] C. Chen, *Robust and optimal control of helicopter systems*, M.Eng. thesis, Nanyang Technological University, 1996

[7] A.B., Chew, *Flight dynamic requirements for helicopter underslung operations*, ALD/RSAF, 1996

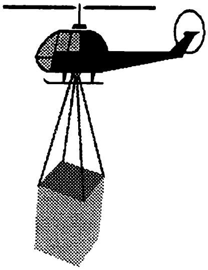


Figure 1: Helicopter Underslung System

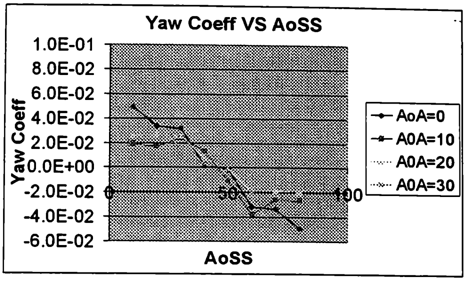


Figure 3: Plot of Yaw Coefficient v.s. AoSS

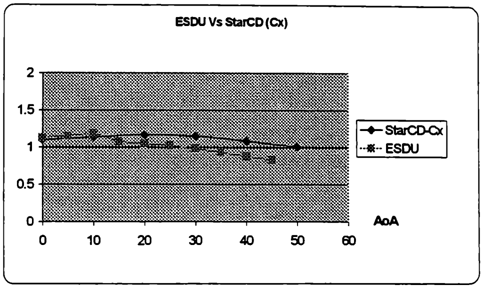


Figure 4: Plot of Cx for StarCD & ESDU

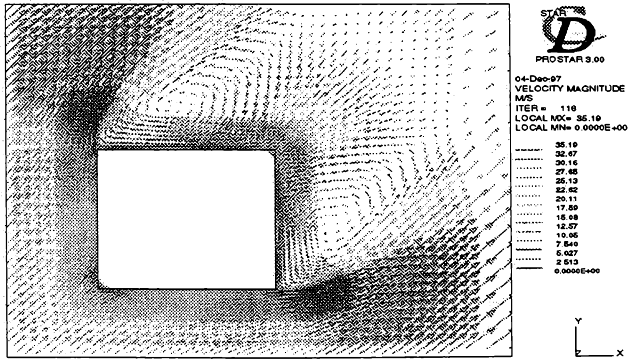


Figure 2: Velocity Magnitude Plot at AoA=50°; AoSS=0°

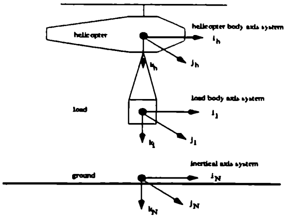


Figure 5: Axis Systems

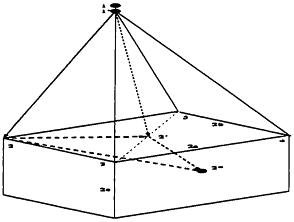


Figure 6: Vectorial Analysis for Underslung Store System

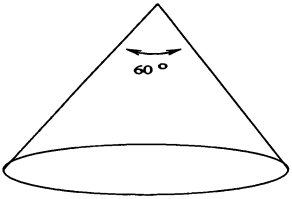


Figure 7: The Store Can Only Oscillate To a Cone with a 60° Top Angle

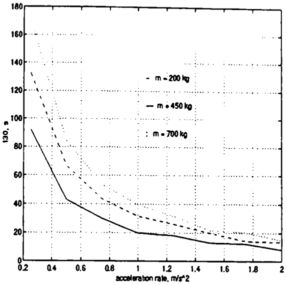


Figure 8: Acceleration envelope for 3 different load mass

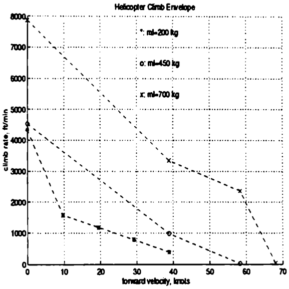


Figure 9: Climb envelope for 3 different load mass

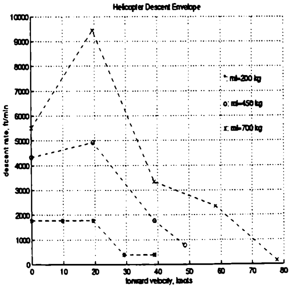


Figure 10: Descent envelope for 3 different load mass

No of cells	117,185
Turbulence Intensity, I	5%
Turbulence Length Scale	0.0096m
Inlet Speed used	30m/s
Density, ρ	1.205 kg/m3
Viscosity, μ	1.81 x 10-5 Pa.S
Thermal Conductivity, k	0.02637 W/m.k
Specific Heat, C_p	1006 J/kg.K
Molecular Weight	28.996
Operating Pressure	1.03 x 105 Pa

Table 1: Fluid Properties

distance between heli. c.g. and attach. point	1.415 m
cable length	3.3 m
cable spring constant	18182 N/m
cable break point	> 18000 N or > 30% elongation
load weight	200 - 700 kg
load length	2.0 m
load width	2.0 m
load height	2.0 m

Table 2: Important parameters used in the simulation

	in-house	$\frac{1}{2\pi}\sqrt{\frac{K}{M_l}}$	relative error
$M_l = 200\ kg$	2.8500 Hz	2.9093 Hz	2.04%
$M_l = 450\ kg$	1.9450 Hz	1.9379 Hz	0.37%
$M_l = 700\ kg$	1.5500 Hz	1.5542 Hz	0.27%

Table 3: Vertical bounce Frequency

cable length	period		relative error
	In-house	NASTRAN	
3.3 m	5.689 s	5.93 s	4.24%
5.0 m	7.314 s	7.34 s	0.36%
10.0 m	10.24 s	10.33 s	0.88%
33.0 m	17.07 s	N.A.	N.A.

Table 4: Periods of pendulum motion

AIAA Paper No. 98-4359

Modeling and Simulation of Flexible Flight Vehicles

David K. Schmidt*

Flight Dynamics and Control Laboratory
University of Maryland
College Park, MD 20742

David L. Raney **

Dynamics and Control Branch
NASA Langley Research Center
Hampton, VA 23681

Abstract

The effects of flexibility on the flight dynamics of aircraft have been shown to be quite significant, especially as the frequencies of the elastic modes become lower and approach those of the rigid-body modes. The handling characteristics of such vehicles are altered significantly from those of a rigid vehicle, and the design of the flight-control system may become drastically more complex. Consequently, the need to accurately model the dynamics of such vehicles, and to develop valid simulations is becoming particularly acute. In this paper the theoretical development of a generic model will be presented. The resulting model structure is appropriate for any flexible atmospheric flight vehicle. Furthermore, the modeling technique used allows for the flexible degrees of freedom to be added to an existing simulation based on a rigid-body model. The data necessary for modeling a specific vehicle includes aerodynamic stability derivatives, aerodynamic influence coefficients, elastic mode shapes, modal frequencies and damping, and generalized masses. An example will be presented based on the B-1 aircraft, which was used to develop a motion-based simulation in NASA Langley's simulation facility. Also to be presented will be examples of the dynamic responses (time and frequency domain) for a generic elastic vehicle. These responses are critical to evaluating hardware and software requirements for simulation fidelity, in terms of visual and motion cues, for example. Furthermore, a brief assessment of the effects of limitations in the simulation will also be presented. Such limitations considered include digital time delay, motion-hardware bandwidth, and motion washout logic.

Introduction

The effects of flexibility on the flight dynamics of aircraft have been shown to be quite significant, especially as the frequencies of the elastic modes become lower and approach those of the rigid-body modes. The handling characteristics of such vehicles are altered significantly from those of a rigid vehicle[1], and the design of the flight-control system may become drastically more complex.[2] Consequently, the need to accurately model the dynamics of such vehicles, and to develop valid simulations is becoming particularly acute.

Shown in Figure 1, for example, are the lowest frequencies of the structural vibration modes for several flight vehicles. This data shows that these frequencies can be lower than 3 Hertz, and in some advanced supersonic-transport configurations (denoted SCR in the figure) the frequencies are as low as one Hertz. Some are well within the bandwidth of the pilot and primary flight-control system, and others may certainly be excited by turbulence. The data also indicates that the flexibility effects may be quite acute for the next-generation supersonic transport vehicle. Figure 2 (from Ref. 1) clearly indicates that as the vibration frequencies are reduced, handling characteristics may be significantly degraded.

In this paper the theoretical development of a dynamic model for a generic vehicle will be presented. The resulting model structure is appropriate for any flexible atmospheric flight vehicle. Furthermore, the modeling technique used allows for the flexible degrees of freedom to be added to an existing simulation based on a rigid-body model. An example will be presented based the generic, large, high-speed aircraft, which was used for motion-based simulations in NASA Langley's simulation facility.

Also to be presented will be examples of the dynamic responses (time and frequency domain) for the generic elastic vehicle. These responses are critical to evaluating hardware and software requirements for simulation fidelity, in terms of visual and motion cues, for example. Furthermore, a brief assessment of the effects of limitations in the simulation will also be presented. Such limitations considered include digital time delay, motion-hardware bandwidth and displacement limits, and motion washout logic.

* Prof. and Director, Assoc. Fellow, AIAA.

** Aerospace Engineer, Member, AIAA.

Background

We are addressing the issue of real-time, manned simulations. Further, these simulations may or may not be motion based. Frequently a 6 DOF simulation of the rigid vehicle has been developed, or the information required (e.g., aerodynamic data) to perform such a simulation is available. So one key issue is how to develop the flexible-vehicle simulation by extending an existing rigid-body simulation.

The information/data that will be required includes the aerodynamic data for the rigid-vehicle, aerodynamic influence coefficients for the elastic deflections, in-vacuo vibration frequencies and modes shapes for the structure, vehicle mass and inertias, and generalized masses for the elastic modes.

Finally, the vehicles for which elastic effects may be significant are of particular interest, for practical reasons. Therefore, we will tend to focus on large, high-performance vehicles. Such vehicles usually have low elastic mode frequencies, approaching those of the rigid-body modes. And this situation is the most acute.

Modeling the Flexible Vehicle

We will assume that a finite-element analysis, and perhaps ground vibration test have been performed, such that a modal description of the structure is available. That is, if the elastic deformation at a position (x,y,z) on the structure is denoted as, then it may be written as

$$\bar{d} = \sum_{i=1}^{\infty} \bar{\phi}_i(x,y,z)\eta_i(t) \qquad (1)$$

where η as a generalized coordinate of the structure, and $\phi(x,y,z)$ is a mode shape, or eigenfunction, associated with the generalized coordinate. Of course, a finite number of modes must be used, so the model always is based on a truncated-mode description.

The selection of which modes to use is critical. But basically one must selected modes that can be excited, and can contribute to important displacements in the system, such as cockpit accelerations. One must also include modes than my significantly couple with the rigid-body motion. All these factor usually implies two to four of the lowest frequency modes, depending on their mode shapes.

The equations of motion are developed with the use of a special body-fixed axis. This axis is referred to by Milne[3] as the "mean axis." This axis is that axis with respect to which the elastic deformations contribute no translational or rotational momentum to the total for the vehicle. Or, for the mean axis; the following relations hold.

$$\int_V \frac{\delta\bar{p}}{\delta t}\rho \; dV = \int_V \bar{p}\times\frac{\delta\bar{p}}{\delta t}\rho \; dV = 0$$

where an infinitesimal volume of the vehicle dV has location $\bar{p}$ with respect to the instantaneous center of mass of the vehicle. All the equations of motion will be derived using this body-fixed axis, and all forces, moments, and inertias will be resolved into components along and about the three directions (i.e., x, y, z) of this axis.

Now the position of dV relative to the cm was denoted as $\bar{p}$. And let $\bar{p} = \bar{s} + \bar{d}$, where $\bar{s}$ is the position of dV with no elastic structural deformation, and $\bar{d}(x,y,z)$ is the displacement of dV due to elastic deformation. Using the modal representation for the elastic deformation $\bar{d}$, the practical mean axes constraints, written below, may be used to actually determine the location of the mean axis if a modal representation of the structure is available (e.g., from a finite-element analysis).

$$\sum_{i=1}^{\infty} \frac{d\eta_i}{dt}\int_V \bar{\phi}_i\rho \; dV = \sum_{i=1}^{\infty} \frac{d\eta_i}{dt}\int_V \bar{s}\times\bar{\phi}_i\rho \; dV = 0$$

Now let the inertial position of the instantaneous center of mass (cm) be R_0, the angular velocity of the vehicle-fixed (mean) axis relative to the inertial frame be $\bar{\omega}$, the total vehicle mass be M, and the vehicle's inertia tensor be I. With the mean axis as defined above, it can be shown[4] that the kinetic energy can be expressed as follows.

$$T = \frac{1}{2}M\frac{d\bar{R}_0}{dt}\cdot\frac{d\bar{R}_0}{dt} + \frac{1}{2}\bar{\omega}^T[I]\bar{\omega} + \int_V \frac{\delta\bar{p}}{\delta t}\cdot\frac{\delta\bar{p}}{\delta t}\rho \; dV$$

Also, using the fact that the first mass moment about the cm is zero, we have for the gravitational potential energy

$$\begin{aligned} U_g &= -\int_V \left(\bar{R}_0 + \bar{p}\right)\cdot\bar{g}\rho \; dV \\ &= -\bar{R}_0\cdot\bar{g}\int_V \rho \; dV - \bar{g}\cdot\int_V \bar{p}\rho \; dV \qquad (2) \\ &= -\bar{R}_0\cdot\bar{g}M \end{aligned}$$

Noting that the generalized mass of the ith mode is defined by

$$M_i = \int_V \bar{\phi}_i \cdot \bar{\phi}_i \rho \; dV$$

the kinetic energy becomes

$$T = \frac{1}{2} M \frac{d\bar{R}_0}{dt} \cdot \frac{d\bar{R}_0}{dt} + \frac{1}{2} \bar{\omega}^T [I] \bar{\omega} + \frac{1}{2} \sum_{i=1}^{\infty} M_i \dot{\eta}_i^2 \qquad (3)$$

Taking the origin of the fixed-body (mean) axes to be at the cm, let

$$\bar{R}_0 = x\hat{i} + y\hat{j} + z\hat{k}$$

and

$$\frac{d\bar{R}_0}{dt} = \frac{\delta \bar{R}_0}{\delta t} + \bar{\omega} \times \bar{R}_0 \equiv U\hat{i} + V\hat{j} + W\hat{k}$$

This relation defines the velocity components U, V, and W. Here again, the three unit vectors are coincident with the orthogonal directions (i.e., x, y, and z) of the body-fixed (mean) axis, and are of course not inertial.

Similarly, we define the three components of the angular velocity resolved in the body-fixed axis by the relation

$$\bar{\omega} = p\hat{i} + q\hat{j} + r\hat{k}$$

where these components are related to the rates of change of the Euler angles by the usual relationships,

or
$$p = \dot{\phi} - \dot{\psi}\sin\theta$$
$$q = \dot{\psi}\cos\theta\sin\phi + \dot{\theta}\cos\phi$$
$$r = \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi$$

Application of Lagrange's equation to the energy terms defined above, (Eqns. 2 and 3), leads to the following elastic airplane equations of motion.

$$M\left[\dot{U} - rV + qW + g\sin\theta\right] = Q_x$$

$$M\left[\dot{V} - pW + rU - g\sin\phi\cos\theta\right] = Q_y$$

$$M\left[\dot{W} - qU + pV - g\cos\phi\cos\theta\right] = Q_z$$

$$I_{xx}\dot{p} - \left(I_{xy}\dot{q} + I_{xz}\dot{r}\right) + \left(I_{zz} - I_{yy}\right)qr + \left(I_{xy}r - I_{xz}q\right)p + \left(r^2 - q^2\right)I_{yz} = Q_{\phi_B}$$
$$I_{yy}\dot{q} - \left(I_{xy}\dot{p} + I_{yz}\dot{r}\right) + \left(I_{xx} - I_{zz}\right)pr + \left(I_{yz}p - I_{xy}r\right)q + \left(p^2 - r^2\right)I_{xz} = Q_{\theta_B}$$
$$I_{yy}\dot{q} - \left(I_{xy}\dot{p} + I_{yz}\dot{r}\right) + \left(I_{xx} - I_{zz}\right)pr + \left(I_{yz}p - I_{xy}r\right)q + \left(p^2 - r^2\right)I_{xz} = Q_{\theta_B}$$
$$\ddot{\eta}_i + \bar{\omega}_i^2 \eta_i = \frac{Q_{\eta_i}}{M_i} \qquad i = 1, 2, 3, \ldots$$

where the terms $Q_{(\cdot)}$ denote the generalized forces acting on the systems, due to aerodynamic and propulsive effects.

Now define the total aerodynamic and propulsive forces and moments, resolved into components in the mean axes as

$$\bar{F} = \mathbf{X}\hat{i} + \mathbf{Y}\hat{j} + \mathbf{Z}\hat{k}$$

$$\bar{M} = \mathbf{L}\hat{i} + \mathbf{M}\hat{j} + \mathbf{N}\hat{k}$$

where

$$\mathbf{X} = L\sin\alpha - D\cos\alpha\cos\beta + S\cos\alpha\sin\beta + T_x$$

$$\mathbf{Y} = - D\sin\beta - S\cos\beta + T_y$$

$$\mathbf{Z} = - L\cos\alpha - D\sin\alpha\cos\beta + S\sin\alpha\sin\beta + T_z$$

and L = lift (aero + prop), D = drag (aero + Prop), and S = lateral force (aero + Prop), **L** = rolling moment, **M** = pitching moment, and **N** = yawing moment.

Then the virtual work done by the aerodynamic and propulsive forces and moments can be written as

$$\delta W = \mathbf{X}\delta x + \mathbf{Y}\delta y + \mathbf{Z}\delta z$$
$$+ \left[\mathbf{L} + (y\mathbf{Z} - z\mathbf{Y})\delta\phi_B\right] + \left[\mathbf{M} + (z\mathbf{X} - x\mathbf{Z})\right]\delta\theta_B$$
$$+ \left[\mathbf{N} + (x\mathbf{Y} - y\mathbf{X})\right]\delta\psi_B + \int_S \bar{P}(x,y,z) \cdot \sum_{i=1}^{\infty} \bar{\phi}_i \delta\eta_i dS$$

So then $Q_x = \mathbf{X}$, $Q_y = \mathbf{Y}$, $Q_z = \mathbf{Z}$, and $Q_{\phi_B} = \mathbf{L}$, $Q_{\theta_B} = \mathbf{M}$, $Q_{\psi_B} = \mathbf{N}$, and the development of the equations of motion is complete. The resulting state vector is

$$\text{state-vector} = [U, V, W, p, q, r, \eta_1, \eta_2, \ldots \dot{\eta}_1, \dot{\eta}_2, \ldots]^T$$

Of course, actuation and sensor dynamics may be appended as necessary, utilizing Eqn. 1 to include elastic displacements.

Force and Moment Modeling

Now consider the issue of modeling the forces an moments. These forces and moments may take on the form below, as shown in Ref. 4. For the lift L, we have

$$L = \frac{\rho Vel^2 S_{ref}}{2} \left\{C_{L_{rigid}} + C_{L_{flex}}\right\}$$

where, for example,

$$C_{L_0} + C_{L_\alpha}\alpha + C_{L_\delta}\delta + \frac{\bar{c}}{2Vel}\left(C_{L_q}q + C_{L_{\dot\alpha}}\dot\alpha\right)$$

and

$$C_{L_{flex}} = \sum_{i=1}^{n}\left\{C_{L_{\eta_i}}\eta_i + \frac{\bar{c}}{2Vel}\left(C_{L_{\dot\eta_i}}\dot\eta_i\right)\right\}$$

Now the rigid-body lift coefficient may be obtained any way that data would be obtained for a normal simulation. The new data that must be generated are the coefficients $C_{L_{\eta_i}}$ and $C_{L_{\dot\eta_i}}$. These would be obtained, for example, from an aeroelastic code, or from analytical expressions as developed in Ref. 4, for example. Finally, analogous expressions would be used for the drag D and moments (e.g., **M**). The key point is that the effects of elastic deflections on the aero forces and moments are captured through the terms above such as $C_{L_{flex}}$.

In addition, the generalized forces Q_{η_i} exciting the elastic degrees of freedom (the η_i's) must also be developed. These generalized forces may also be expressed in terms of non-dimensional coefficients similar to those below.

$$Q_{\eta_j} = \frac{\rho Vel^2 S_{ref}}{2}\left\{C^{\eta_j}_{rigid} + C^{\eta_j}_{flex}\right\}$$

where, for example,

$$C^{\eta_j}_{rigid} = C^{\eta_j}_{0} + C^{\eta_j}_{\alpha}\,\alpha + C^{\eta_j}_{\beta}\,\beta + C^{\eta_j}_{\delta}\,\delta$$
$$+ \frac{\bar{c}}{2Vel}\left(C^{\eta_j}_{p}\,p + C^{\eta_j}_{q}\,q + C^{\eta_j}_{r}\,r\right)$$

and

$$C^{\eta_j}_{flex} = \sum_{i=1}^{n}\left\{C^{\eta_j}_{\eta_i}\,\eta_i + \frac{\bar{c}}{2Vel}\left(C^{\eta_j}_{\dot\eta_i}\,\dot\eta_i\right)\right\}$$

Note now that the effects of the rigid-body motion (e.g., α, q) on the elastic deflections $\phi_i\,\eta_i$ are captured through the above relations. In other words, the above coefficients couple α, q, etc. to the elastic degrees of freedom. All the above coefficients are obtained from an aeroelastic code, or expressions such as those in Ref. 4.

Example Vehicle

Consider now a generic, swept-wing vehicle, with a conventional empanage. The descriptive data is given below in Table 1. We will include in the analysis the longitudinal dynamics, and will include the first four symmetric vibration modes of the structure in the mathematical model. All the aerodynamic coefficients for the elastic vehicle are listed in Table 2, taken from Ref. 4. Finally, the mode shapes are given in the same reference.

Table 1. Geometry, mass, and inertia of study vehicle

Geometry	$\bar{c}$ = 15.3 ft (mean chord)
	b = 70.0 ft (wing span)
	S = 1946 ft^2 (planform area)
	Λ = 65 degree (sweep angle)

Weight	W = 288,017 lb

Inertia	I_{xx} = 950,000 slug-ft^2
	I_{yy} = 6,400,000 slug-ft^2
	I_{zz} = 7,100,000 slug-ft^2
	I_{xz} = - 52,700 slug-ft^2
	$I_{xy} = I_{yz} = 0$

Modal generalized masses	M_1 = 183.6 slug-ft^2
	M_2 = 9586.5 slug-ft^2
	M_3 = 1334.4 slug-ft^2
	M_4 = 43,596.9 slug-ft^2

Modal vibration frequencies	ω_1 = 12.6r/sec
	ω_2 = 14.1 r/sec
	ω_3 = 21.2 r/s
	ω_4 = 22.1 r/sec

Modal damping ζ_i = 0.02, i = 1, 2, 3, 4

A time history of the pitch-rate response, measures at the cockpit, to elevator input for this generic vehicle is shown in Fig. 3. Also shown in the figure is the same response if the vehicle were modeled as a rigid body. Clearly the elastic contribution to this response is significant.

Similarly, the pitch-rate-to-elevator frequency response for the rigid and elastic vehicle model is shown in Fig 4. Note the short-period modal frequency near 2 rad/sec, and the first aeroelastic modal frequency near 2 Hz are evident.

Table 2. Total force coefficients for baseline study vehicle

Q_x	Q_z	Q_{θ_B}	$Q_{\eta}i$	i = 1	i = 2	i = 3	i = 4
C_{X_0} -0.028	C_{Z_0} -0.34	C_{M_0} -0.252	$C_0^{\eta_i}$	0.0	0.0	0.0	0.0
C_{X_α} 0.0035	C_{Z_α} -0.051	C_{M_α} -0.029	$C_\alpha^{\eta_i}$	-2.6e-04	4.5e-04	2.6e-04	5.84e-07
$C_{X_{\dot\alpha}}$ 0.0	$C_{Z_{\dot\alpha}}$ 0.0	$C_{M_{\dot\alpha}}$ -4.3	$C_{\dot\alpha}^{\eta_i}$	0.0	0.0	0.0	0.0
C_{X_q} -1.7	C_{Z_q} 14.7	C_{M_q} -34.75	$C_q^{\eta_i}$	-9.49e-02	1.16e-02	3.97e-02	2.83e-05
C_{X_δ} 0.0027	C_{Z_δ} -0.0076	C_{M_δ} -0.045	$C_\delta^{\eta_i}$	-2.24e-04	-1.12e-03	4.46e-04	2.56e-06
$C_{X_{\eta_1}}$ 0.0	$C_{Z_{\eta_1}}$ -0.0288	$C_{M_{\eta_1}}$ -0.0321	$C_{\eta_1}^{\eta_i}$	5.85e-05	4.21e-03	2.91e-04	2.21e-05
$C_{X_{\eta_2}}$ 0.0	$C_{Z_{\eta_2}}$ 0.306	$C_{M_{\eta_2}}$ -0.025	$C_{\eta_2}^{\eta_i}$	-9.0e-05	-9.22e-02	1.44e-03	-1.32e-04
$C_{X_{\eta_3}}$ 0.0	$C_{Z_{\eta_3}}$ 0.0148	$C_{M_{\eta_3}}$ 0.0414	$C_{\eta_3}^{\eta_i}$	3.55e-04	1.97e-03	-3.46e-04	9.68e-06
$C_{X_{\eta_4}}$ 0.0	$C_{Z_{\eta_4}}$ -0.0140	$C_{M_{\eta_4}}$ -0.0183	$C_{\eta_4}^{\eta_i}$	1.20e-04	3.37e-03	1.44e-04	1.77e-03
$C_{X_{\dot\eta_1}}$ 0.0	$C_{Z_{\dot\eta_1}}$ -0.0848	$C_{M_{\dot\eta_1}}$ -0.159	$C_{\dot\eta_1}^{\eta_i}$	-4.20e-04	8.71e-03	-6.29e-04	5.55e-05
$C_{X_{\dot\eta_2}}$ 0.0	$C_{Z_{\dot\eta_2}}$ 1.03	$C_{M_{\dot\eta_2}}$ 1.23	$C_{\dot\eta_2}^{\eta_i}$	-1.97e-04	-2.98e-01	1.95e-02	4.09e-04
$C_{X_{\dot\eta_3}}$ 0.0	$C_{Z_{\dot\eta_3}}$ 0.0608	$C_{M_{\dot\eta_3}}$ 0.172	$C_{\dot\eta_3}^{\eta_i}$	6.50e-04	4.19e-03	-8.77e-06	-4.66e-05
$C_{X_{\dot\eta_4}}$ 0.0	$C_{Z_{\dot\eta_4}}$ -0.0199	$C_{M_{\dot\eta_4}}$ -0.0496	$C_{\dot\eta_4}^{\eta_i}$	-1.40e-04	4.15e-03	-1.31e-03	7.99e-02

Note: α and δ in degrees, q and $\dot\alpha$ in rad/s, and η_i in rad, and $\dot\eta_i$ in rad/s.

Sample Results

A dynamic aeroelastic simulation model such as the one described above was implemented in NASA Langley Research Center's Visual-Motion Simulator [5]. This motion-based simulation facility provided acceleration cues to the pilot which represented the aircraft flight dynamics including aeroelastic vibration effects. Six test pilots were asked to compare maneuvers performed with and without aeroelastic dynamic effects present in the real-time simulation model. The pilots' Cooper Harper Ratings (CHR) of a lateral-offset landing maneuver are shown in Table 3. The offset landing task is a challenging maneuver which requires the pilot to aggressively correct for a 300-ft lateral offset from the runway centerline at an altitude of 250 ft. Results indicate that the presence of dynamic aeroelastic effects in the simulation model greatly degraded the aircraft flying qualities, particularly in the lateral axis. In some cases pilots changed their lateral/directional CHR from Level I to Level III ratings as a result of the aeroelastic effects.

Table 3. Impact of Aeroelastic Effects on Test Pilots' Cooper-Harper Ratings

	Longitudinal CHRs						Lateral/Directional CHRs					
Pilot	A	B	C	D	E	F	A	B	C	D	E	F
ASE OFF	3	4	4	5	4	5	3	3	3	4	4	5
ASE ON	6	7	6	7	5	6	4	7	8	6	5	7

Pilot comments indicated that cockpit vibrations due to aeroelasticity degraded the flying qualities ratings for at least two subtly different reasons. The first is that the vibration environment simply had a negative impact on the comfort level or ride quality at the pilot station. Pilots therefore increased Cooper Harper Ratings because the extreme vibrations tended to increase their perception of workload.

Pilots also remarked that cockpit vibrations tended to influence the precision of their control inputs. Some pilots indicated that the vibrations actually resulted in involuntary control inputs. This aeroelastic effect is referred to as *Biodynamic Feedthru*, and was alluded to in [6]. In some cases, the combination of the aeroelastic aircraft, the control stick, and the pilot's biomechanical dynamics may result in a closed-loop system which is unstable or lightly damped. In such instances, cockpit vibrations may cause resonance of the pilot's biodynamic frame, resulting in sustained feedthru of aeroelastic vibrations back into the control stick, a condition which will be referred to as *Biodynamic Coupling*. An analytical model of a similar coupling phenomenon was presented in [7] based on the analysis of flight data.

Figure 5 presents an analysis of a lateral-offset landing task in which the pilot experienced biodynamic coupling while flying the aeroelastic configuration. Frequency and time data shown in these figures have been normalized as follows:

f_o = Normalization Frequency, corresponds to peak in voluntary pilot input frequency spectrum obtained from PSD of pilot stick time histories

T_o = Normalization Time Step, $1/f_o$

The time history at the top of the figure shows lateral cockpit accelerations in g's (dashed line) and lateral stick deflections (solid). Although the units on the two quantities differ, the scaling of ± 1 is convenient since it represents the maximum throw for lateral stick deflection and since lateral g's commanded by the simulation remained in the range of ± 1 g. The plot in the lower left of Figure 5 shows the power spectral density of lateral accelerations and lateral stick deflections applied to a segment of the time history. The frequency spectrum of the pilot's control inputs during this period lies within the pilot's voluntary input bandwidth. The frequency spectrum of the lateral accelerations at the pilot station shows some content at the first and second antisymmetric mode frequencies due to minor turbulence excitation of these structural modes.

The power spectrum of a later segment of the time history is shown in the lower middle of Figure 5. This plot indicates the bulk of the pilot's input spectrum remains in his voluntary frequency band, but it also shows some frequency content of the pilot's inputs in the range of the lateral elastic modes. Once the pilot begins to move the stick at the resonant frequency of the first antisymmetric structural mode there is tremendous potential for the lateral mode to be excited by the control inputs, producing larger lateral accelerations at the pilot station. These lateral accelerations move the pilot's frame in a fashion which produces involuntary control inputs that further excite the structural mode. The third power spectrum plot at the lower fight of Figure 5 covers the final segment of the time history. Here, the spectrum of the pilot's stick input exhibits a pronounced resonant peak at·the frequency of the first antisymmetric structural mode. It is highly unlikely that the pilot's inputs in this frequency range are voluntary. Video of the seated pilot clearly depicted a correlation between lateral stick inputs and involuntary lateral motions of the pilot's upper body. A clear change in the character of the pilot's stick inputs is apparent in the time history, indicating well-developed biodynamic coupling as lateral accelerations feed through the pilot's frame and back into the control inceptor. The pilot could break the involuntary coupling loop if he released the stick, but he is approaching the flare and therefore is unwilling to do so.

A number of similar incidents of biodynamic coupling were encountered when test pilots flew the aircraft model with structural modes present in the simulation. Encounters with biodynamic coupling were always dangerous and sometimes catastrophic. At least three of the six test pilots encountered biodynamic coupling during some portion of the experiment. Examples of such incidents for pilots B, E, and C are shown in the Figure 6. Power spectra of the pilots' stick inputs for each of these cases indicate a resonant peak at the frequency of the first antisymmetric elastic mode.

Very little biodynamic coupling was encountered with the symmetric elastic modes. The lateral axis is far more prone to such coupling for a number of reasons. First, the pilot's seat tends to support the body longitudinally and vertically, but not laterally, so side-to-side accelerations are more difficult to resist. Furthermore, symmetric modes produce vertical accelerations while the stick input is fore and aft, so there is less tendency for the pilot's body motions to feed directly into the stick. However, antisymmetric modes produce lateral accelerations which feed directly into lateral stick deflections. A sidestick control inceptor was used in this experiment. The susceptibility of various inceptor types to biodynamic feedthru is a potential topic for future investigations.

To summarize, biodynamic coupling is indicated when cockpit vibrations due to elastic modes feed directly through the pilot's arm and back into the control stick, creating a lightly damped or unstable closed-loop system. The phenomenon is evidenced by a resonant peak in the power spectrum of the pilot's stick inputs at the frequency of one or more of the dynamic elastic modes. The tendency to couple with structural modes appears to increase when pilots tighten their grip on the stick, often in preparation for the flare as the aircraft nears the runway. The phenomenon is influenced by design of the control inceptor and control laws, piloting style and probably even various aspects of the pilot's physical stature. These results highlight the importance of modeling and simulation of aeroelastic effects when assessing the flight dynamics and flying qualities of large flexible aircraft.

Demands on the Simulation Facility

Figure 7 was taken from a 1973 report which documented the frequency response capabilities of the Langley Visual-Motion Simulation Facility [5]. The input/output amplitude ratios for vertical and lateral sinusoidal inputs of 1.8 inches are shown, along with the resulting phase lag, for input frequencies from 0.1 to 12 rad/sec. Also shown on these plots is the frequency range of the dynamic elastic modes that were included in the simulation experiment described above. It is clear that the dynamic elastic portions of the model reside at the upper threshold of motion platform's capabilities. The motion base appears to possess reasonable performance at the lowest elastic mode frequency range indicated in Figure 7 (0.8 amplitude ratio vertically and 1.0 amplitude ratio laterally, with about 15 deg of phase loss.) For the aeroelastic simulation study described above, only the six lowest frequency elastic modes were retained in the model (3 symmetric and 3 antisymmetric.)

Figure 8 provides a comparison of vertical accelerations at the pilot station that were commanded by the real-time simulation (dashed line) with those that were actually produced by the motion platform as measured by accelerometers (solid line). The time history was taken from a landing task that was performed with dynamic aeroelastic effects present in the model in which the pilot experienced biodynamic coupling. The plot at the top of the figure shows that the magnitude of the two signals compared favorably, with the motion platform sometimes delivering vertical acceleration cues as high as 0.4 and 0.6 g's. The two segments at the bottom of the figure provide a closer look at the frequency content, amplitude ratio and time delay between the two signals. The gross frequency content of the signals appears quite similar, although a time delay is apparent between the command and the actual measured acceleration. This delay is approximately equal to 0.06*To. The accelerometer measurements shown in the small-amplitude excerpt at the bottom left of the figure appear to contain an uncommanded high frequency component at many of the inflection points in the time history. The cause of this small-amplitude aberration is uncertain, but it is most likely due to mechanical "slop" in the structural components of the motion platform or the mounting of the accelerometer package, itself. This vibration is present throughout the time history, but its magnitude is small in comparison to the commanded accelerations.

Figure 9 provides a comparison of lateral accelerations at the pilot station that were commanded by the real-time simulation (dashed line) with those that were actually produced by the motion platform as measured by accelerometers (solid line). The comparison of the lateral accelerations appears similar to the previous comparison of the vertical acceleration. The two segments at the bottom of the figure provide a closer look at the frequency content and time delay between the two signals. A time delay is apparent between the command and the actual measured acceleration which is slightly longer than that which was observed in the vertical axis. The accelerometer measurements shown in the small-amplitude excerpt at the bottom left of the figure again contain an uncommanded high frequency component at many of the inflection points in the time history. The amplitude ratio between the actual and commanded lateral accelerations is approximately 0.8. The extent to which these motion fidelity limitations impacted the assessment of dynamic aeroelastic effects is thought to be minimal, but their presence should be kept in mind when interpreting the results of the simulation.

The measured accelerations of the simulator cab indicate that the motion base provided a reasonable representation of the dynamic aeroelastic model in terms of frequency content and magnitude of the motion cues. As aircraft designs become larger and more flexible, the need for improved motion-based simulation facilities capable of accurately representing elastic effects increases in significance.

Summary/Conclusions

In this paper we addressed the need for accurate modeling and high-fidelity simulation of highly flexible aircraft. Experimental results obtained from motion-based piloted simulations of a generic vehicle clearly demonstrated the significant impact that dynamic aeroelasticity can have on the flying qualities of a large flexible aircraft. The simulation also revealed the potential for a dangerous form of dynamic interaction referred to as biodynamic coupling. The results highlight the need for motion-based simulation facilities capable of accurately representing elastic effects so that the flight dynamic and flying qualities characteristics of such aircraft may be more clearly understood.

Acknowledgments

The first author would especially thank Mr. Sang Lee, for generating a significant portion of the numerical results presented.

References

1. Waszak, M. R., Davidson, J. D., and Schmidt, D. K., "A Simulation Study of the Flight Dynamics of Elastic Aircraft," NASA Contractor Report 4102, Vols. I and II, Dec., 1987.

2. Schmidt, D. K, "Integrated Control of Flexible High-speed Aircraft.," AIAA Guidance, Navigation and Control Conference, Baltimore, MD, August 1995

3. Milne, R. D., "Dynamics of the Deformable Airplane," Parts I and II, Her Majesty's Stationary Office, Reports and Memoranda No. 3345, Sept. 1962.

4. Waszak, M.R. and Schmidt, D.K., "On The Flight Dynamics of Aeroelastic Vehicles," *Journal of Aircraft*, Vol. 25, No. 6, June, 1988.

5. Parrish, Russell V.; Dieudonne, James E.; Martin, Dennis J.; Copeland, James L.: "Compensation Based on Linearized Analysis for a Six-Degree-of-Freedom Motion Simulator". NASA TN D-7349, 1973.

6. Ashkenas, I.L.; Magdaleno, R.E.; McRuer, D.T.: "Flight Control and Analysis Methods for Studying Flying and Ride Qualities of Flexible Transport Aircraft". Systems Technology, Inc.; Contract NAS1-1847, NASA Contractor Report 172201, August 1983.

7. Smith, John W.; Montgomery, Terry: "Biomechanically Induced and Controller-Coupled Oscillations Experienced on the F-16XL Aircraft During Rolling Maneuvers", NASA TM 4752, Dryden Flight Research Center, Edwards, CA, 1996.

Trends in Elastic Frequencies

Aircraft	Frequency (r/s)
B-1	13
Concorde	13+
C-5A	11.
NASP	~18.
SCR designs	~6.5

Figure 1, Examples of Lowest Structural Vibration Frequencies

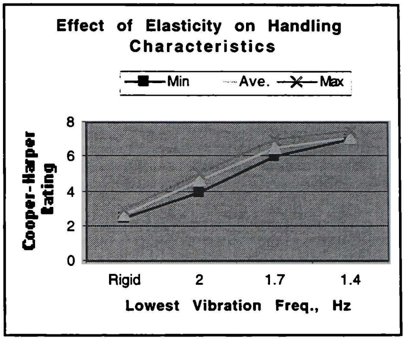


Figure 2, Effect of Increased Flexibility on Handling Characteristics (Cooper-Harper Rating)

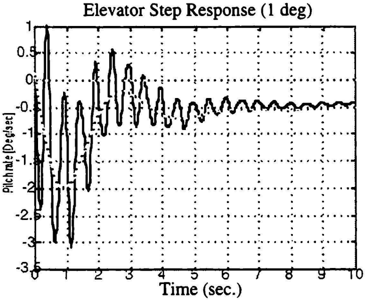


Figure 3, Attitude Time Response

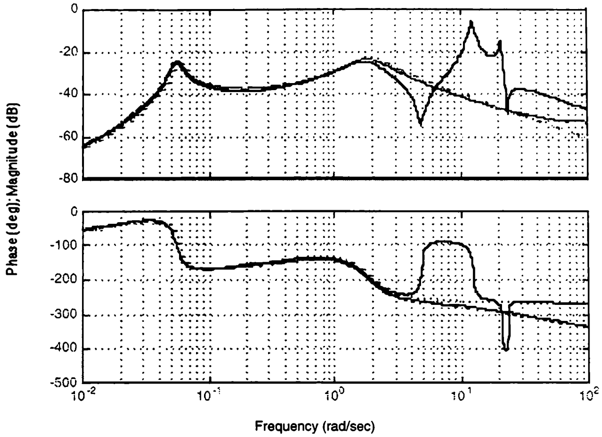
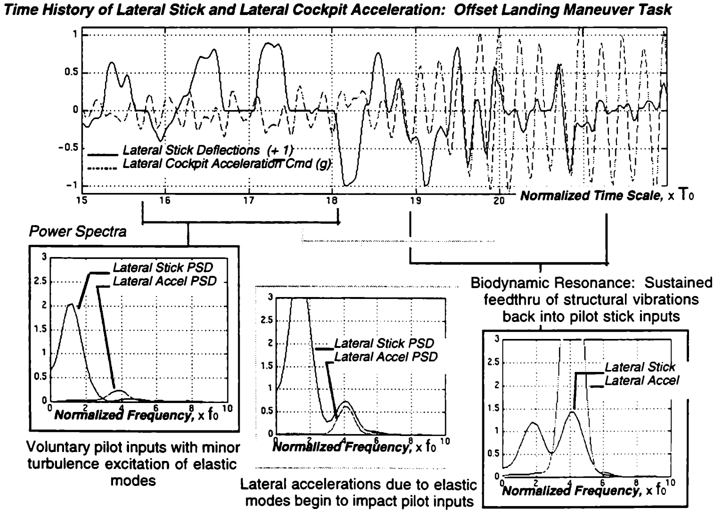


Figure 4, Pitch rate to elevator, Mag (dB) and Phase (deg)
Flexible and Rigid Vehicle


Time History of Lateral Stick and Lateral Cockpit Acceleration: Offset Landing Maneuver Task

Voluntary pilot inputs with minor turbulence excitation of elastic modes

Lateral accelerations due to elastic modes begin to impact pilot inputs

Biodynamic Resonance: Sustained feedthru of structural vibrations back into pilot stick inputs

Figure 5, Example of Biodynamic Coupling Incident

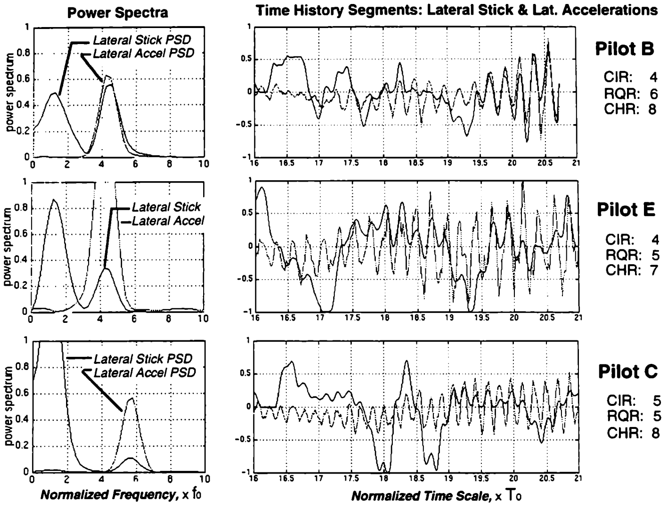


Figure 6, Biodynamic Coupling Incidents for 3 Pilots

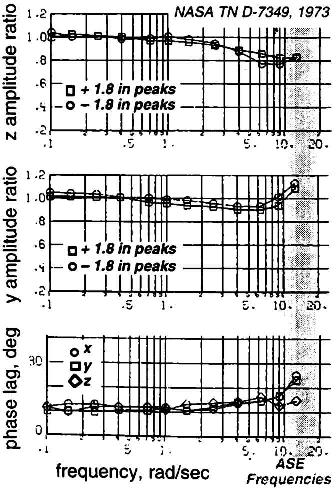


Figure 7, Frequency Response of Langley Visual/Motion Simulator

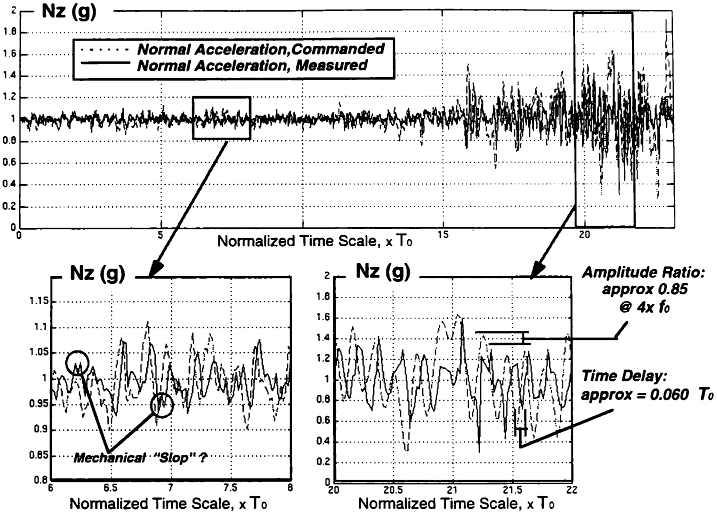


Figure 8, Motion Platform Response: Nz@ Pilot Station

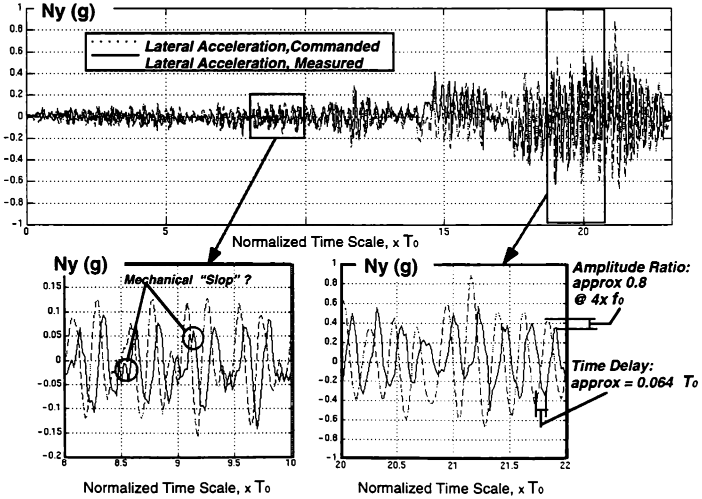


Figure 9, Motion Platform Reponse: Ny @ Pilot Station

AIAA-98-4360

DYNAMIC MODELING
FOR ACTIVE VIBRATION REDUCTION IN AEROSPACE VEHICLES

Khanh Ly, PhD.
GEC-Marconi-Aerospace-Inc.
New Jersey 07981-1640

ABSTRACT

Techniques of dynamics modeling for an Active Vibration Reduction System in aerospace applications are discussed. Flights are steadily effected by internal as well as by external disturbances. Typically, the internal disturbances are changes in control surface deflection, changes in center of gravity, flaps, landing gear or wing sweep angle. External disturbances are usually generated by atmosphere turbulence, upset guts, altitude, temperature. Although airplanes are always designed to oppose or to diminish the motions resulting from such disturbances, the built-in tendency to resist can be augmented. Such disturbances should not endanger the operation of the airplane or result in serious discomfort for its passenger. A desired vibration environment in the aircraft can be achieved by using an Active Vibration Reduction System with force generators that react actively against the vibration sources. Each force generator will use its own inertia to produce the commanded force at its attachment point in the aircraft body, by exerting equal and opposite contact forces between two points in the fuselage. The design justification is derived from a simplified computer model, especially configured to demonstrate response to transient commands and required input current. The computer model is basically a position servomechanism. The model simulation represents the performance of the final system, and therefore it is an evaluation for accuracy, settling time and power requirements.

PRINCIPLE

The objective of vibration control system is to isolate the flexible aerospace payload from any excitation during flight. Control of low as well as high frequency excitations is neccessary to ensure the vibration-free environment required for flights. There have been considerable efforts to control vibration of large, flexible space structures under various loading conditions (Reference 1).

When flight conditions of the aircraft change, the vibration environment in the fuselage will change. In order to adapt to the new conditions, the system controller must send a new command signal to the force generator. The command signal, that corresponds to the vibrations, has the form of a cosine function with three signal parameters: amplitude, frequency and phase (Reference 2). The transient performance of force generator is designed under the consideration of step changes in amplitude, phase or frequency individually or simultaneouslly, all three at the same time.

Inertial reactions of airplane structure are the most important criteria to designers at the beginning of development phase. The integrity of structure due to dynamic aeroelasticity is effective only at high frequencies. The short-term behavior of structural displacements is still often neglected. Solutions of aeroelastic problems are confronted between high and lower structural frequencies (Reference 3).

The vibration reduction in aerospace vehicles can be carried out by using passive damping devices, which cover a range of expensive processes, for example, using highly demanded material to keep a minimum weight or smart structures to absorb the stimulus. In general, a passive damping device is very effective at high frequency (>500 Hz) where the dimension of the device is comparable with the wavelength of vibration (Reference 4). Besides the passive devices are usually bulky and room-consuming. The most important

In contrast, active vibration damping device is usually more effective at lower frequencies (<500 Hz), typically in the range from 10 to 20 Hz (Reference 5). Many studies and tests have shown that active systems are the most competent. A significant advantage of active vibration damping devices is in application for increasing the flight comfort to helicopter crew and passenger. In addition, the weight was decreased dedicated to the active devices.

In Reference 6, a Higher Harmonic Control System showed that remarkable vibration reductions could be achieved by using active systems. The vibration information of the blade passage frequency was obtained by sensors on the airframe. An onboard computer calculates the vibration frequencies and sends command signals to anti-vibration actuators. The actuators oscillate harmonically to provoke an opposing load to the vibrations and thereby an active damping of the vibration in the fuselage is generated.

Regarding application arrangement, there are two different types of actuators for active vibration control: linear precision actuators and active isolation fittings. Linear precision actuators apply frequency-dependent damping force to the structure by accelerating a mass mounted on an energized device. While linear precision actuators apply external, inertial loads to the structure, active isolation fittings are intrastructural devices that can expand or contract to absorb the motion of the base structure without transmitting the motion to the payload. These devices reduce vibration both at major resonances of the system and between resonances (Reference 7). They operate as primary load-bearing parts of the structure and alter the structural dynamics to drastically reduce the transmission of the vibration to the isolated body (Reference 8).

Recently, new technology has emerged for modeling, analysis and control of structures. These advances were made possible by the development of new materials that are manufactured as thin layers to be used as distributed sensors and actuators. These materials convert their deformations into electrical signals that are almost linear with the strain. Under electrical energy with an electrical signal, the actuators deform and generate forces. Layers of such materials can be embedded between or outside of the existing structures to transform into an actuating device. The most advantage of these actuators are that they produce actuation force by adding just an extra insignificant weight. They are used in form of modal actuators to take the advandtage of the orthogonal eigenfunctions of a structure. Theoretically, they can be designed in any form to adapt to the orthogonality (Reference 9).

With application in active damping systems, smart materials such as piezoceramics and magnetostrictives are designed to change their shape when exposed to an external stimulus (Reference 10). A mass actuator makes use of a material known as Terfenol-D to improve the quality of helicopter rides by reducing vibration and rotor noise. Terfenol-D is a magnetostrictive material that can exhibit large strains when subjected to a magnetic field. The components of magnetostrictive actuators consist of a rod of giant magnetostrictor Terfenol-D surrounded by a solenoid, all of which is encased in a prestressed housing to enhance strain per field performance. Once sensors have picked up the frequencies of fuselage vibration, a voltage is dispatched to the Terfenol-D core, which reacts by expanding. The resulting counter-impact of the reaction mass against the motor housing can cancel the vibrations and the noise can be reduced by 40 dB (Reference 11).

Some other smart materials which have significant success in application for aerospace vibration damping are piezoelectric ceramic, nitinol, nickel titanum. Among these materials, piezoelectric actuators, which the author worked with for long time (Reference 12), showed a very reliable response to be effectively integrated in a digital control loop. At the same design envelope, piezoceramic (PZT- lead zirconate titanates) actuators can supply a specified damping level in a shorter time interval with significantly less energy than the piezopolymer (PVDF- polyvinylidene fluoride) actuators. On the other hand, PVDF actuators are lighter in weight but stiffer.

In general, active vibration damping systems have significant advantages in comparison to passive systems:

- reduce aircraft vibration more effectively and over a greater range of flight conditions
- have lower weight
- consume less energy
- have more reasonable life cycle cost
- no single point failure by using the availabe redundancy in flight control system

The time required to achieve a desired damping level decreases with more actuators. A reduction of actuators used in a system would challenge the performance requirements and can remark an evaluation of a design depth. One of the key design point is that damping the disturbances at their sources requires fewer actuators than those, once they have already propagate thoughout the system (Reference 13). For structures with complexity of mass, high load or stiffness properties, active vibration control with smart materials is still not yet developed for pratical applications. At first, due to the multiple modes needed to be controlled, the controller for each mode must be stacked which causes an undesirable arrangement (Reference 14). Therefore its application is limited by the magnitude of the vibration. Besides, its sucess is also limited by the high bandwidth the actuation system is trying to control. Because of a finite charge tolerance, the piezoelectric can not supply high loading, so they can only control a limited structural damping. Finally, their overall actuation force is too low and neglible compared to the mechanical force generation discussed in this paper.

The basic principle behind active vibration control is simple: The vibration is detected using a sensor and the signal is inverted, amplified and fed back to an actuator which acts in an opposite sense, reducing the vibration. In Figure 1 and 2, the concept of the active vibration damping system is illustrated. Figure 1 shows the vibration signals and the generated signals, where the generated signals are not perfectly inverted from the vibration signals because of some phase delays. Figure 2 demontrates the result signals of vibration reduction.

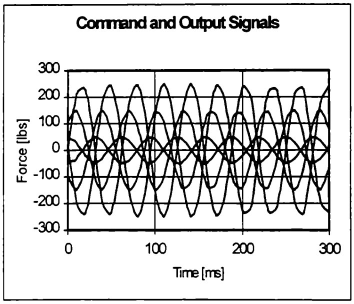


Figure 1

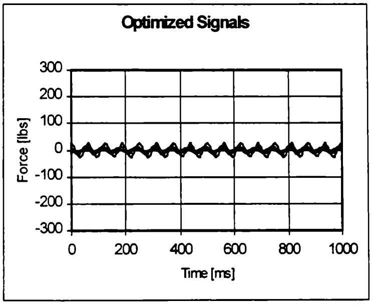


Figure 2

There are different methods to build a vibration damping systems (Reference 15). The simplest method is the isolation of the vibration source. In an airplane, the most severe vibration source is the engine. The engine produces vibration which is transmitted throughout the structure. In order to isolate the vibration source, the engine is bolted on an active vibration damping platform without any rigid joints to the rest of the aircraft structure (Reference 16). This method is effective, because the vibration is reduced at source.

The most commonly used method is to localize the vibration, regardless of the vibration source. The anti-vibration actuators generate the opposing loads locally on most critical places, where vibration expected to be at maximum. This method is easy to apply but because of its local solution, the overall performance is less than the first method.

A more effective method is to use a central control unit to collect all the vibration data, for example, in the cabin of an airplane. Then all the command signals are accordingly calculated and sent to appropriate opposing actuators.

In order to generate an opposite force, the command signal is tranfered to the two motors in two components. Briefly, the frequency of the command signal will control the speed of both motors and the amplitude of signal will determine the phase shift between the two motors, in order to modulate the required force vector. In each motor, the phase is shifted with the same value but in different direction of rotation. It means, the phase shift enables an advance motion for the angular position of the one motor and a retard motion for the other motor at the same time.

The principle of such a phase shift is explained fundamentally by studying the case of modulating a maximum force. To produce a synergized force, the two motors are brought exactly into the same angular position, where each pair of unbalanced gears with their eccentrical masses are counter-rotating and are exactly symmetrical to the common tangent.

MODELING

The computer model is basically a position servomechanism. The following model analysis will represent the performance of the final system.

The Command Signal

Adapting to the steadily varying flight conditions and environmental vibration, the system controller send command signals to the device for anti-vibration generation. The command signal, that corresponds to the vibrations, has the form of a cosine function:

$$a = A\cos(\omega t + \varphi) \tag{1}$$

where A [Volt]: amplitude of the command signal
ω [rad/s]: frequency of the signal
φ [rad]: signal phase.

The three signal parameters A, ω, φ are variable, depending on the type of vibration. The command signal should change in the frequency envelope of a given frequency range, in the amplitude envelope of up to 5 volts and in the phase envelope of 180 to -180 degrees. The relation of the value of frequency f and ω in (1) is given by:

$$\omega = 2\pi f \tag{2}$$

The transient performance of the force generator are designed under the consideration of step changes in amplitude, phase or frequency individually or simultaneouslly, all three at the same time.

Generation of the opposing force begins when the command signal is tranfered to the motors. Then the frequency of the command signal will control the speed of motors and the RMS value of the signal will determine the phase shift between the motors. A combination of these parameters will modulate the required force vector.

The modulation of phase shift is based on the principle of superposition of force vectors to achieve a maximum force. A synergized force is created if all motors are brought exactly into the same angular position, where all the actuators will produce the opposing force in the same direction, in case of a maximum opposing force is required.

If this maximum force is to be reduced in order to adapt to the vibration force, the present angular positions are shifted from each other with the same phase but in different directions of rotation. In this way the synergy effect will be totally canceled out when the phase shift makes a flat line: $\pm 90^0$. In this case, there is no force generated because the inertia forces lie in opposite directions.

The electronic controller receives the change of the command signal and translates it into the phase shift command for the motor. To simulate a change in amplitude of the command signal, a step change of amplitude from 0 to 5 volts is sent to the motor signal. It is expected, that the force generator remains in zero modulation if the amplitude of command signal is zero and the force genarator must reach the maximum modulation if the amplitude of the command signal increases to its maximum value of 5 volts. In this way the motor signals between the two motors has a phase difference of π in case of zero modulation and no phase lag in case of maximum modualtion. In order to simulate the maximum modulation from zero, the phase difference γ between the two motors is determined in accordance with the value of the command amplitude:

$$\gamma = \cos^{-1}(a/A) \tag{3}$$

where a is the considered amplitude at present and A is the possible maximum amplitude. Depending on an advance or retard effect, this step change will be added or subtracted from the motor signal. A phase limiter helps to keep safely the phase shift for each motor between $\pm\pi/2$ to produce a total phase shift of π between two motors.

Mechanical Modeling

In mechanics, there are always 4 categories that a frictional contact can be expected for any mechanical system.
1) No interface displacement between the surfaces: drive surfaces such as power belts, tires on road or clamped surfaces such as press-fitted pulley on shaft.
2) High friction to absorb mechanical energy such as brake discs or clutches.
3) Constant friction to control the quality such as the metal thickness in sheet metal rolling mill
4) Low friction to increase efficiency such as in gears and bearings.

The basic equation in elementary mechanics is:

$$F = \mu N \tag{4}$$

where:

μ:	Friction coefficient
N:	pressure
F:	resulting friction

Material and pattern of friction surfaces is a major factor in the design and prediction the stability and life of the system. The friction element generates heat and unstability due to sliding of metal surfaces (Reference 17).

In general, friction coefficient in equation (4) is a non-linear function of load, speed, lubricant viscosity, local temperature. The general purpose of lubrication is to provide a protective film to the solid surfaces, preventing cohesion and reducing friction, heat by acting as an interfacial layer of low shear strength. Five different forms of lubrications can be identified: hydrodynamic, hydrostatic, elastohydrodynamic, boundary and solid film.

In case of rotating bearings with hydrodynamic lubrication, the asperities are separated by a thick film. The pressure is built up by the relative moving between the two surfaces into a wedge-shaped zone, so that it can maintain an adequate amount of fluid between the two surfaces. The traction of the lubricant is predictable. The stability can be obtained and explainable by the laws of fluid mechanics. Hydrodynamic lubrication will become elastohydrodynamic lubrication, when contact between the surfaces occurs. The traction of fluid can still somehow predictable by Hertzian contact stress and fluid mechanics.

The instability and turbulences can be resulted, when the film is so thin and the surface area is so small that the pressure between surfaces is not sufficient to separate them. In general, a mixed lubrication between hydrodynamic, elastohydrodynamic type occurs. In contrast to the hydrodynamic lubrication where the viscosity of the lubricant is important, the chemical composition dominates the stability and the shear capability during vibration. Under extremely high acceleration pressures, the bearing sufaces get in contact and create a short-term boundary lubrication, which can cause an instability of the system with direct intermetallic interactions.

Newton defined the viscosity as "the resistance which arises from lack of slipperiness in a fluid ...".

$$\sigma = \eta \frac{d\gamma}{dt} \tag{5}$$

where:

$\sigma = \frac{F}{A}$:	Shear Stress
$\frac{d\gamma}{dt} = \frac{v}{h}$:	Shear rate
v:	velocity
h:	film thickness
η:	viscosity

It is reported in many researches that this equation is valid only for film thickness more than 50 molecules. The viscosity increases when the film gets thinner. The viscosity becomes solid when the film becomes so thin to be anisotrop surface, less than a few molecular layers.

Reynolds described in 1886 the phenomena of hydrodynamic pressure, which is built up by load and relative sliding speed of two surfaces. Under pressure lubricant molecules seem to stretch out in strata parallel to the solid boundaries. Closest to the boundaries, molecule density is lowest because the can not be located precisely. So the density in the next neighbor area would be very high. The density wave propagates outward and flattens out when film thickness reaches about 10 Angstrom. Between two solid boundaries, the oscillation caused by changing molecule density creates a disorder of the tendency to the strata parallel organization of molecules.

The same phenomena happens in directions parallel to surfaces. In general, the wave of density disorder propagates laterally and radially. In the direction tangent to the surfaces, the wave extends along the boundaries and outward from the boundaries. Finally, it tends to form a decaying oscillation. There are three typical characteristics: spatial and temporal variations in the density and pressure of the lubricant film, asperity deformation and cavitation caused by asperity collision.

A non-linear viscosity results when the structure of the lubricant is disturbed. This happens when the relaxation is very low caused by pressure and sliding speed. It would be more severe when the friction is integrated in a servo loop, where the control command reacts sensitively to a frictional feedback. Any additional hysteresis on the feedback path will magnify the deviation of the expectancy and feedback signal.

For applications, where extreme temperature is required such as in aerospace or military, the volatility is the most important factor to maintain the stability of system of the high end of the wide spectrum, for example from -65^{0}C to 150^{0}C and above. At high temperature, a superior oxidation and thermal stability deteriorate and degenerate if the lubricant begins to evaporate.

On the other side, at extreme low temperature, the lubricant becomes more viscous, the lubricating film gets solider to be sheared by the same motive force. The force driver must be able to produce extra force necessary to overwhelm the increased shear reaction of the lubricant film.

The stability of the system can only maintained if the lubrication is consistent over a long period of time and a wide range of temperature. In addition, in order to keep a lowest degradation of system performance due to stability, a zero lubricant separation or a maximum consistency is also required for long-term application or endurance operation under high temperature.

A simplified model of the system for force generation can be described with the viscous damping and traction, that is usually accepted as friction, within the hydrodynamic and elastohydrodynamic lubrications. Under these mixed viscous conditions, the calculated traction depends on the pressure or load on the bearing surfaces.

The pressure equation in hydrodynamics is given by:

$$\frac{h}{R} = \frac{4.896U\eta}{W} \tag{6}$$

where:

h:	film thickness
R:	boundary curvature
U:	velocity
η:	atmospheric viscosity
W:	load per unit width

And the pressure in elastohydrodynamics can be obtained by the equation

$$\frac{h}{R} = \frac{2.65\left(\frac{U\eta}{ER}\right)^{0.7}(\alpha E)^{0.54}}{\left(\frac{W}{ER}\right)^{0.13}} \tag{7}$$

where:

h:	film thickness
R:	boundary curvature
U:	velocity
η:	atmospheric viscosity
α:	pressure viscosity coefficient
E:	Elastic modulus
W:	load per unit width

During vibration the bearing surfaces separated from each other and then joined together again to build up a kind of self-induced vibration. In the equation for elastodydrodynamics, the pressure is not as important as the speed of rotors. This explained why the force generation is more effective at higher frequency.

Electrical Modeling

Under normal condition, the angular position θ [Rad] of a motor is a function of speed:

$$\theta = \theta_0 + \omega t$$

(8)

where t : time [sec]
ω: the speed [Rad/s]
θ_0: the position [Rad] at t=0

The electrical and speed torque characteristic performance of brushless DC motor can be described as:

$$V = IR + L_s\, dI/dt + K_b d\theta/dt \tag{9}$$

where V [Volts]: the terminal voltage across the active commutated phase
I [Amps]: the sum of the phase current into the motor
R [Ohms]: the equivalent input resistance of the active commutated phase
L_s [Henries]:the equivalent input inductance of the active commutated phase
K_B [Volts/Rad/sec]: the back EMF constant of the active commutated phase
$d\theta/dt$ [Rad/sec]: the speed of motor
t [sec]: time

In the simulation model, Voltage Limiter and Current Limiter are necessary to cut off the higher amplitudes to a maximum allowable limit, so that a damage to the motors is avoidable. These limits are the maximum voltage and maximum current that cause a maximum temperarature rise (ambient and internal heating) permitted by the motor manufacturer.

If K_T is the torque sensitivity [Ncm/Amp], the developed torque T [Ncm] is proportional to the current I:

$$T = K_T I \tag{10}$$

Consideration of the dynamic performance of motor under load, a dynamic mechanical relation between torque T and angular position θ is obtained:

$$T = (J_M + J_L)\, d^2\theta/dt^2 + F_d d\theta/dt + T_F + T_L \tag{11}$$

where J_M [kgcm^2]: the moment of inertia of motor
J_L [kgcm^2]: the moment of inertia of the gears
T_F [kgcm^2]: the motor friction torque
F_d [Ncm/Rad/sec]:the damping factor
T_L [kgcm^2]: the load torque

For standard gravity operation, the gravity factor K_g = 1. In order to simulate the worstcase of flight condition, the gravity factor is chosen as K_g = 2.5, which will affect the driving torque of the motor.

SIMULATION MODEL

The performance of system is simulated based on a model using SIMULINK, a MATHLAB application software. The mechanical characteristics is derived and simplified from a real prototype, which was built in accordance with requirements for a real environment. In general, the actual performance depends mostly on the technical parameters of the motors. The most important part of an aerospace application is the design of an optimized solution for a possible light-weight motor. The accuracy and the reliability of the simulation model is improved by interchange between simulation results and real test data from the prototype.

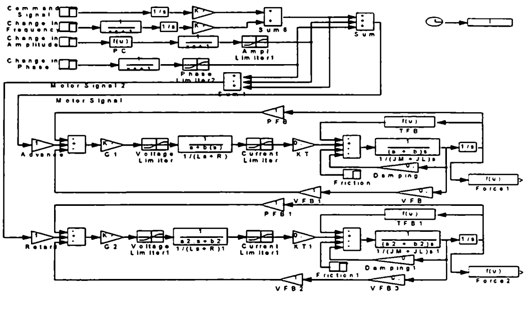


Figure 3

RESULTS

The figure 4 shows performance relating to a simultaneous step changes in amplitude, phase and frequency of the command. In order to examine the worst case of the anti-vibration system, the command signals for worstcase are simulated to study the limitations of system response. The worstcase of the phase change together with the worstcase of frequency change at the same time can be considered as the worstcase of any arbitrary simultaneous step change in command signal. The most arbitrary change in command signal occurs when the frequency increases from the minimum to maximum frequency, the phase varies from zero to 180 degrees and the amplitude increases from 0 to 5 volts. All changes take place at the same time t=3 seconds. The change of frequency is seen by the change of wavelength. The motor can maintain steady state performance after 0.5 seconds.

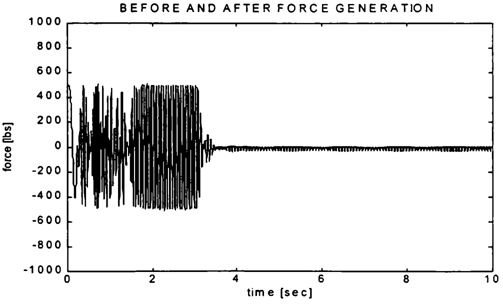


Figure 4

In the first 1.5 seconds from turn-on, there is large initial transients because of the inertia load of itself and of the system. After about 2 seconds, the motor is fully under control. It exists large initial turn-on transients, but will not occur in the real system since the PLL will initially bring the unit up to speed gently.

An interesting point of this total change in command signal amplitude is the total response of the generated force. The generated force drops clearly from the maximum modulation to the zero modulation at time 3 sec. The zero modulation of the force generator corresponds to the zero amplitude of the command signal. At time 3 sec, in order to response to the minimum amplitude of the command signal, the zero modulation of the force generator cuts off an opposing force of 500 lb to achieve a zero vibration status and waits for the next signal.

CONCLUSIONS

An analysis of the system dynamics using simulation technique was discussed. The interaction between digital signal processing and mechanics as well as electronics was mentioned, so that a mathematical model can be built based on kinetic characteristics, electrical energy and inertia loads. The results of simulation were compared to the measurements on the test prototype.

The force generation is in form of a module with motors driving opposing actuators. A vibratory force is generated along the common axis with the other. In combination it produces a total force of 500 pounds unidirectional. Two units of force generators can exert a maximum force of 1000 pounds. In the test data measured on a real prototype, it showed no significant difference between generated force obtained mathematically by simulation and force measurements collected by sensors on a real physical unit.

Because there is no significant difference between simulation model and prototype, the advantages of simulation model against real prototype are:

- Easier to change design parameters
- Faster for verification of system performance
- Time and cost savings

REFERENCES

1) Lee-Glauser G., Ahmadi G.: "Vibration Isolation of Launch Vehicle Payload and its Subsystems", Journal of Aerospace Engineering, January 1995

2) McRuer, D., Ashkenas, I. and Graham, D.: "Aircraft dynamics and automatic control", Princeton University Press, 1973

3) Udwadia F.E., Kalaba R.E.: "Equations of Motion for Mechanical Systems", Journal of Aerospace Engineering, July 1996

4) Ye J.Q., Soldatos K.P.: "Three-Dimensional Vibrations of Crossply Laminated Hollow Cylinder with Clamped Edge Boundaries", Journal of Vibration and Acoustics, July 1997

5) Welsh, W.A., Von Hardenberg: "Test and Evaluation of Fusealge Vibration Utilizing Active Control of Structural Response optimized to ADS-27". Proceedings of the 46th Annual Forum of the American Helicopter Society, May 1990.

6) Achache, M., Polychroniadis, M.: "Higher Harmonic Control Flight Tests of an Experimental System on the SA349", 42nd Annual Forum of the American Helicopter Society, June 1988.

7) Phillips, D.J., King J.A., Davis L.D., Hyland D.C. "High Performance, Active Control of Dynamically Complex Space Structures", Journal of Guidance, Control and Dynamics, September 1996

8) Okuma M., Shi Q.: "Identification of Principal Rigid Body Modes under Free-Free Boundary Condition", Journal of Vibration and Acoustics, July 1997

9) Yang S.M., Sheu G.J., Yang C.D.: "Vibration Control of Rotor Systems with Noncollated Sensor/ Actuator by Experimental Design", Journal of Vibration and Acoustics, July 1997

10) Jones, L.D., Garcia E.: "Self-Sensing Magnetostrictive Actuator for Vibration Suppression", Journal of Guidance, Control and Dynamics, January 1996

11) Leventon, W.: "Aircraft Design the Smart Way", page 42, PD&D April, 1997

12) Ly, Khanh: "Verfahren zur makro und mikroskopischen Topographie mit Digitaler Bildverarbeitung", Stuttgart, Germany, May 1994

13) Panza M.J., McGuire D.P., Jones P.J.: "Modeling, Actuation and Control of an Active Fluid Vibration Isolator", Journal of Vibration and Acoustics, January 1997

14) Callahan J., Baruh H.: "Active Control of Flexible Structures by Use of Segmented Piezoelectric Elements", Journal of Guidance, Control and Dynamics, July 1996

15) Aplin, J.: "Active noise control from research to reality", Digital Avionics Systems Conference, Phoenix, Arizona, November 1994

16) Magnus, K.: "Schwingungen", B.G.Teubner, Stuttgart, Germany, 1969

17) Ly, Khanh: "Chatter of Dynamic Brake: Numerical Model vs Experimental Prototype", Proceedings of the 68th Shock and Vibration Symposium, Huntville, Maryland, October 1997

AIAA-98-4361

MULTIBODY SIMULATION IN THE INTEGRATED DESIGN OF SEMI-ACTIVE LANDING GEARS

W.R. Krüger, W. Kortüm
DLR - Deutsches Zentrum für Luft- und Raumfahrt
Institut für Robotik und Systemdynamik
D - 82234 Wessling

Summary

The paper presents a technique for an integrated aircraft - landing gear design using multibody simulation (MBS). The MBS technique is aimed at the simulation of the dynamics of a technical system in its entity. It has proven to be a well balanced compromise between the requirements of "fast", "robust", and "exact" simulation, thus assisting the process of virtual prototyping.

An aircraft model is set up using SIMPACK, DLR's prime MBS tool. The aircraft model presented is based on a large civil aircraft consisting of a finite element structure for fuselage and wings, landing gear models from CAD, control structures from a control design tool, and measured input data for several runway profiles.

The objective is to achieve an optimal landing gear design with respect to the aircraft as a whole. Therefore, a multi-objective optimization loop is used to optimize landing gear parameters using criteria covering both the landing and rolling performance, i.e. resulting loads, stress and comfort at several locations at the aircraft structure.

In addition to the conventional suspension design, this method has been used to propose a control strategy for semi-active damping of the nose landing gear that reduces runway induced vibrations of fuselage and wings, thus increasing comfort and minimizing airframe loads.

1. Landing Gear Requirements

The importance of the landing gear as an integral system of the aircraft is often underestimated. Take-off and landing are among the most crucial moments during a flight, and a short-range aircraft with several flights a day can accumulate up to 500000 km on the ground in its lifetime. Additionally, the landing gear is one of the few aircraft systems without redundancies. Therefore a fail-safe design is essential.

Thus, the landing gear is more than just a set of wheels. It is a sophisticated device that has to fulfill several, sometimes conflicting requirements. The function which is most evident is the absorption of energy, both vertical (by means of the shock absorbers) and longitudinal (by means of the brakes). This requirement has led to the American term "landing gear". The English expression "undercarriage" emphasizes the second important task of the system, the provision of means for the aircraft to navigate around the airfield for taxi and take-off, at the same time providing a smooth ground ride to pilot and passengers, [1], [2].

Since the second world war those requirements have been met at most commercial aircraft by the nose gear tricycle wheeled configuration equipped with an oleo-pneumatic shock absorber, the so-called "oleo". The landing gear has become a complex system that leaves only limited room for improvement. Certification requirements and geometric conditions concerning the location of the landing gears further restrict the design freedom, [4].

There are, however, innovative approaches that promise to improve the landing gear performance without compromising weight and safety. Two of them will be presented in this paper: *integrated design strategies* link the development of landing gear and aircraft structure so that the properties of the complete system are taken into consideration from the earliest steps of the design process onwards; furthermore, *semi-active landing gears* may solve the inherent comfort problem of an aircraft rolling over a rough runway with a landing gear optimized for a hard landing impact.

2. Landing Gear Design

2.1 Traditional Design Methods

The design of a new aircraft takes a long time span which can last up to a decade from the first proposals to customer delivery. The development process includes different manufacturers and suppliers who provide many components to the aircraft. As a rule, at an early design stage, the locations of the gears and the number and size of the tires are defined to form a basic landing gear layout. A specialist for the development and the manufacturing of the gears is chosen who receives the landing gear specifications including aircraft mass and geometric aircraft data along with the operational envelope (e.g. aircraft speed, and whether the aircraft has to operate from a paved or an unpaved airfield). From that point on, aircraft and airframe are, in the majority of cases, developed and optimized independently.

Conway, [5], and Currey, [6], provide standard procedures for landing gear design. Frequently, landing gears for civil aircraft are also dimensioned by certification requirements, namely the FAR 25 and JAR 25, [7]. These requirements determine that a landing gear must be able to resist a free drop with a vertical touch down velocity of 3.05 m/s (10 fps) under a mass representing the vertical static aircraft weight plus an additional component for the pitching moment at touch-down. Loads representing towing and landing with a roll angle are usually defined as load factors.

The certification loads are assumed to be conservative, i.e. an aircraft is unlikely to encounter such conditions often in its life time. Still, since it is mandatory that a landing gear meets the requirements, the FAR 25 forms a starting point for both the design of the mechanical parts and the adjustment of the shock absorbers.

The shock absorbers, together with the tires, form the main element for the vertical dynamics of the landing gear. The two main factors that can be varied by the designer are the gas spring volume and pressure and the oil damping properties. They are optimized such that the loads in the certification drop test mentioned above are kept to a minimum.

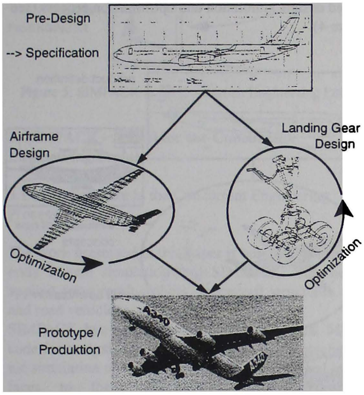


Figure 1: Conventional Aircraft / Landing Gear Design

Parallel, the aircraft manufacturer designs the airframe according to ground load estimations, again from relatively simple models. Finally, the landing gears and the airframe are assembled for the prototype which then begins with the test flights (see figure 1).

2.2 Problems of the Traditional Approach

The design process described in section 2.1 leads to landing gears which meet the certification requirements and perform well on the test rig. There is, however, no certainty that the system "aircraft / landing gear" as a whole also possesses satisfactory system dynamics. A landing gear might induce higher loads into the airframe than foreseen. Aircraft flexibilities might lead to fuselage oscillations at taxi and take-off. Models of single landing gears or rigid body aircraft might not reveal some important problems of the aircraft at operation. Figure 2 shows an example case. The system response of a large transport aircraft encountering two bumps of a height of 3.8 cm (1.5 in), 21 m (70 ft) apart is calculated using a rigid aircraft model (case 1). The resulting vertical cockpit acceleration reaches a level of 0.2 g. Using an aircraft model with elastic airframe representation, however, the resulting acceleration reaches 0.6 g, three times the value found for the rigid aircraft. Obviously the bumps excite an aircraft fuselage mode, a result which is in accordance with observations at real aircraft. The rigid model proves to be inadequate to detect potential problems of this sort. This may lead to an unsatisfactory design requiring high re-engineering costs. This can be avoided if the dynamics of the complete aircraft can be evaluated before the prototype is built.

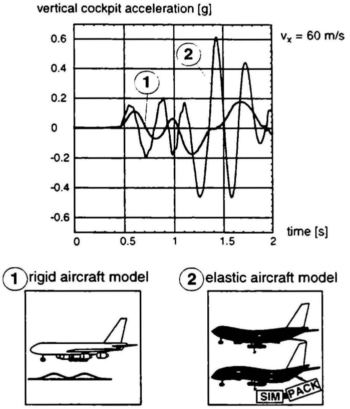


Figure 2: Taxi Over Double Bump

2.3 Concurrent Engineering Methods in Landing Gear Design

Tests on life-size aircraft are obviously expensive and risky, and experiments on test-rigs (namely drop-test facilities) allow only limited deduction of information about the landing gear dynamics; especially the interaction between aircraft and landing gear is difficult to assess. On the other hand, simulation offers a means to examine the behavior of the airplane as a complex system in its environment at a reasonable cost. This paper presents a method for integrated design based on simulation using the multibody simulation (MBS) approach.

Multibody simulation has demonstrated to be a simulation technique which is well-suited for this kind of design and analysis. With a modern computer mechanical systems with a large number of degrees of freedom, both for rigid and elastic body motion, can be modeled. Multibody simulation combines the ability to simulate models including arbitrary non-linear motion and highly non-linear force laws and with powerful analysis methods and a fast model set-up using components from libraries and from neighboring engineering disciplines. The latter becomes more and more important as the continuous and close interaction of all disciplines and partners involved in the development of aircraft becomes increasingly intensified. The procedure of simultaneous multi-disciplinary work throughout the design process is known under the name of *concurrent engineering (CE)*, [8].

Many engineering disciplines, e.g. aerodynamics, computer aided design, finite element analysis and control design software are based on the analysis of simulation models. Thus, a large number of aircraft models are set up in parallel during the design phase. One of the main tasks of the concurrent engineering process is

- to aid the engineer in setting up his problem-specific model using the knowledge and model components created by experts in related disciplines and
- to transfer the results of a group to other disciplines in an automized way to ensure that all models include the relevant changes at minimum effort for model update.

Figure 3 shows the classical relationship between the knowledge about a technical system and the effort and costs it takes to implement modifications to the system design. Regarding the fact that the time scale for product development will tend to ever shorten in the future it is evident that an early integration of the engineering disciplines and the corresponding model update has to start as early as possible.

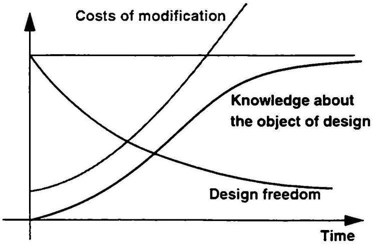


Figure 3: System Knowledge vs. Design Freedom [9]

Next to the large number of co-existing simulation models the aircraft industry faces another problem: research, design, and construction of aircraft are spread among a number of companies on a variety of sites, even in different countries. There are many subcontractors, research institutes, and universities in the aeronautical field. This multitude poses, of course, some problems: experts often cannot work in a team; know-how is classified for a certain company or institution; work is done redundantly; company culture can differ quite strongly; software tools are incompatible, and a common data base is missing.

To extract the advantages from a multi-site, multi-company situation, the concurrent engineering approach includes a number of measures: the creation of a common data base with access for all partners at the necessary level; well-defined interfaces between the software tools; a steady automized model transfer in an integrated design, simulation and optimization process (see figure 4).

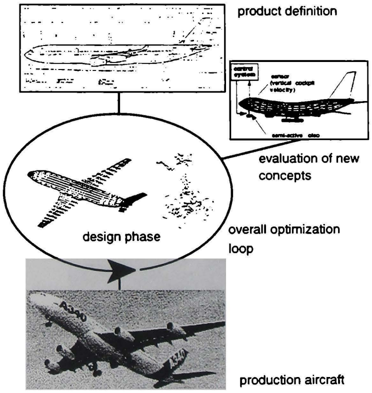

product definition
control system
sensor (vertical cockpit velocity)
semi-active oleo
evaluation of new concepts
design phase
overall optimization loop
production aircraft

Figure 4: Integrated Aircraft / Landing Gear Design

Multibody simulation can be the kernel of a powerful design package including tools from CAD, FEA, and control design if the appropriate interfaces are available, [10]. It can combine the functionalities of the specialized tools for a fast and thorough design analysis. The main advantage is that multibody simulation allows a good compromise between the requirements of high model accuracy at a moderate model complexity as well as low computation times and a large number of analysis methods both in the frequency and the time domain.

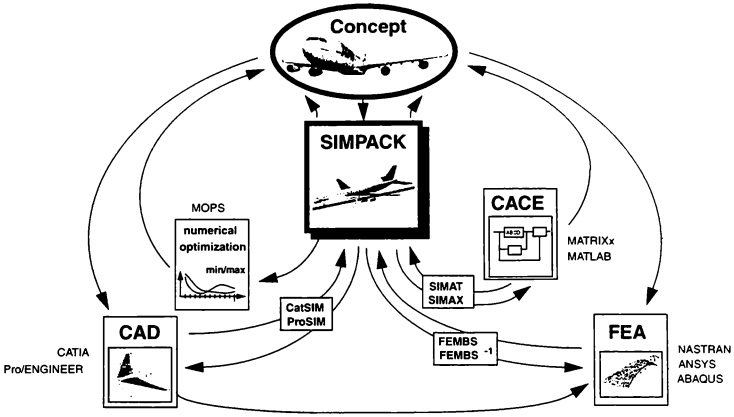


Figure 5: SIMPACK in the Concurrent Engineering Loop

3. SIMPACK - A Tool for the Concurrent Engineering Process

3.1 MBS Software in the Concurrent Engineering Process

The work presented in this paper is based on the multibody system simulation tool SIMPACK. It is being applied for aircraft, robots, spacecraft structures, rail and road vehicles.

SIMPACK has been developed from a typical MBS-code for analysis of specified systems into a mechatronic simulation and design tool with bi-directional interfaces to the major CAE (Computer Aided Engineering)-software packages. The basis of SIMPACK as of any MBS-software are its *multibody formalisms*, i.e. the algorithms which automatically generate the equations of motion. In SIMPACK, *O(N)*-algorithms - the number of operations grows only linearly with the number of the degrees-of-freedom - are establishing the equations of motion in explicit or in residual form, [11]. The equations of motion for the MBS are set up in the form of ordinary differential equations (ODE) or - particularly in the case of closed-loops - optionally in differential-algebraic form (DAE). Numerical integration algorithms for time integration were incorporated or developed, some of them in close correlation with the formulation of the equations of motion.

Beyond the solvers for time-integration (i.e. the narrow sense of simulation) a variety of special numerical analysis methods, in particular for *linear system analysis* (linearization, eigenvalues, root locii, frequency response, stochastic analysis in time- and frequency-domain) were modified for their special use in vehicle dynamics and integrated. Moreover, computational procedures for stationary solutions (equilibria, nominal forces) and for kinematic analysis were developed and implemented.

Extensive libraries of coupling elements like joints and force elements as well as excitations aid the engineer in setting up a model. This includes the kinematics of different suspension systems or other complex joints, force models for hydraulic components, various tire models etc.; subsystem modeling techniques enable the user to establish complex non-standard kinematic and force laws.

3.2 Interfaces

One of the most important features of SIMPACK is the fact that SIMPACK is an *open* system which possesses various interfaces to external standard software products (see figure 5) from the domain of finite element analysis (FEA), computer aided design (CAD), and control system analysis (CACE) programs.

FEA:

For the examination of elastic vibration control it is of course necessary to use elastic instead of rigid bodies in the model set-up. In SIMPACK the kinematics of elastic bodies as well as stress stiffening effects of elastic deformations are taken into account. To read in arbitrary flexible geometry a file interface to FEA-programs was developed. This gives access to mass, stiffness and damping matrices and to the above mentioned terms of second order. Vice versa, loads computed by SIMPACK can be transferred to the FEA-code to be used in stress analysis.

CAD:
For the purpose of incorporating physical and graphical CAD-data, SIMPACK can be linked by a function call interface to CAD-packages, establishing an addition to the interactive SIMPACK model set-up tool. The link enables data consistency between MBS- and CAD-data at each step of the model definition process.

CACE:
Due to its ability to numerically linearize the system equations, SIMPACK can be used as a linear simulation block within computer-aided control engineering tools. It may also be linked as a fully nonlinear block into nonlinear optimization and control tools like MATRIXX and MATLAB. SIMPACK can also work in co-simulation mode retaining all functionality for model set-up, analysis and model animation.

Code Import / Export:
Fortran or C code can be introduced into the SIMPACK models via the User Function Interface. Inversely, a SIMPACK model can be exported as Fortran or C code to be used in standard applications, problem-specific programs or in simulators.

3.3 Parameter Variation and Optimization (MOPS):

In the system design process it is frequently the case that while the basic layout is defined, a number of parameters are still free. These parameters may include mechanical parameters like attachment locations or shock absorber lengths, force law parameters like spring stiffness or damping coefficients, or control parameters, e.g. gains. As an aid for the design engineer SIMPACK offers an automized parameter variation which is used to get an overview over system performance as a function of parameter changes. Additionally, an optimal configuration can be found with SIMPACK's multi-objective optimization tool MOPS (Multi-Objective Parameter Synthesis), [12]. The simulation evaluated within the optimization loop can include simultaneous static, linear and nonlinear simulations of multiple MBS-models (multi model approach). A data handling module is added for structured control of the interactive optimization design process, where design parameters, model and simulation scenarios are varied, starting out from the first optimization test to the final optimal system, [13].
The basic optimization loop is indicated by figure 6: during the optimization, MOPS supplies the simulation with model parameters, the time integration and the evaluation of the criteria are performed in the MBS program, the criteria are then passed back to MOPS, which in turn supplies the simulation with a new set of parameters.

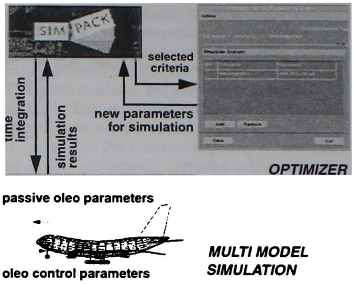


Figure 6: Optimization Procedure

4. Semi-Active Control of Aircraft Landing Gear

4.1 Semi-Active Shock Absorbers

Although passive aircraft suspensions, mostly consisting of a spring and a damping device, today have a high standard, they suffer from a disadvantage that results from their principle. Designed to absorb the energy of a hard landing impact, aircraft suspensions perform quite poorly in reduction of ground-induced loads during taxi and take-off. In a few cases vibrations are known to have become so severe that aircraft had to abolish take-off because the pilot could no longer read his instruments, [14]. Not surprisingly, supersonic aircraft and a new generation of stretched civil transport aircraft tend to suffer most under ground-induced structural vibration because of their increased structural flexibility inherent in their design with slender bodies and, at supersonic aircraft, their relatively thin wings.
First proposals for adaptive landing gears have been made in the seventies, [15], and with the advent of microelectronics both in digital computers and in controller development a new type of controlled landing gear is feasible. Both systems controlling strut stiffness and damping coefficient (fully active) and controlling only the damping parameters (semi-active) have been subjects of research, [16], [17]. Active and semi-active suspensions have found their way into automotive, truck and railway applications.
The usual means of control is a closed-loop control with a control law acting with respect to measured quantities (often velocity or acceleration) at certain points of reference. A suspension termed "semi-active" has the restriction that it cannot input energy into the system. Usually realized as damping control, it is therefore only able to control the amount of energy dissipation. Only forces of the same direction as that of the instantaneous relative damper velocity can be generated. Nevertheless, since fully active control systems

usually require a heavy and costly force generation device (for aircraft landing gears mostly pressurized oil reservoirs have been proposed) they are unlikely to be quickly introduced in production aircraft. Semi-active suspension systems offer considerable advantages of light weight and less complicated mechanical requirements without suffering a great loss in performance, [18], compared to the active shock absorber. Additionally, a semi-active oleo fulfills the requirements of a fail-safe design because at system failure the oleo works as a (non-optimal) fixed orifice shock absorber.

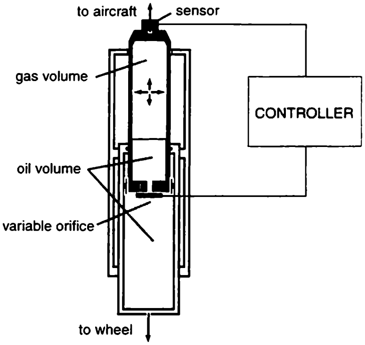


Figure 7: Semi-Active Oleo [20]

4.2 Control Laws:

The main reasons for the introduction of semi-active landing gear control are to minimize the load on the airframe structure, reduce force peak values and vibrations that can reduce the life of the airframe and to minimize accelerations acting on pilot and passengers since the induced vertical and horizontal accelerations and vibrations can lead to passenger discomfort (a question of comfort) and crew disorientation (a question of safety). Significant improvement of this unpleasant situation can be achieved by a control algorithm known as "Sky-Hook" damping, [19]. The vertical aircraft velocity is fed back into the oleo to control the orifice cross-section between the oil chambers at compression and at decompression, which in turn leads to significant reductions in aircraft vertical oscillations as well as of the vertical forces in the landing gears (see figure 8).

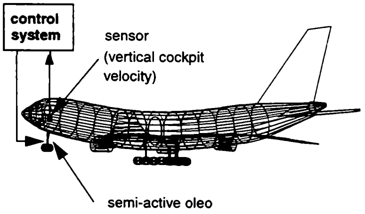


Figure 8: Sky-Hook Damping: Control System Sensor and Actuator Location [20]

5. Integrated Design: Optimization of an Aircraft Nose Landing Gear

5.1 A Concurrent Engineering Case Study

The optimization of a nose landing gear with respect to the aircraft as a whole including a passive and a semi-active shock absorber has been a case study for the integrated aircraft / landing gear design. The project included the airframe manufacturer who supplied the airframe model, the landing gear manufacturer supplying the landing gear models, and DLR using its software to establish an integrated design loop between the partners.

5.2 Model Description

For the optimization of a landing gear the aircraft model has to be complex enough to cover all important effects of the system dynamics in the frequency range of interest but, at the same time, it has to be as simple as possible to allow a fast analysis in the time domain. To correctly represent the aircraft dynamics, fuselage and wings have to be modeled as flexible bodies. Basis for this example is a finite element model with 500 nodes which is able to represent airframe natural frequencies of up to 100 Hz. For this study, the criteria for the landing gear optimization in the taxi mode are the vertical accelerations at several aircraft locations, e.g. at cockpit or center of gravity. Since the objective is to improve comfort for pilot and passengers the decisive frequencies lie in the range of 1 to 10 Hz, [21]. To prevent deterioration of comfort at higher frequencies (the so-called "spill-over"), elastic modes up to 20 Hz are being considered. Due to the symmetric excitation only the symmetric modes with an additional three static modes have been chosen and were included via modal reduction using the Ritz-approach, [22], to be used in the MBS simulation. The resulting structure model uses twelve elastic degrees of freedom. Airframe and landing gears are coupled in three attachment points for each landing gear. These joints are either rigid or consist of a bearing representing the attachment stiffness.

For the landing impact the forces in each landing gear and the attachment points to the aircraft structure are of main interest. The landing gears are rigid body models derived from CAD with non-linear force laws for shock absorbers and tires. The rigid body modeling is sufficient for this load case since the loads at taxi can be considered as mostly vertical and do not lead to lateral or horizontal landing gear oscillation. However, the horizontal landing gear oscillation known as "gear walk" is included. This is especially important for analysis of the landing impact because gear walk plays a major role at tire spin-up and at braking.

The tire model also plays an significant role both for landing and taxiing. The wheel has a rotational degree of freedom, the tire consists of a point follower including a vertical spring with horizontal slip.

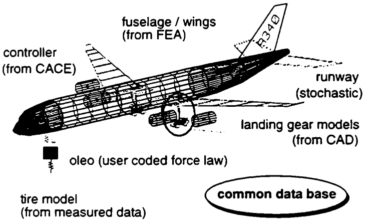


Figure 9: Full Aircraft Model and Used Interfaces to Other CAE Disciplines

Model Simplifications for the Optimization:

The model described so far has been set up in detail for evaluation purposes ("evaluation model"). However, for control layout and optimization some simplifications are applied to reduce the time necessary for the analysis, leading to a "design model". First, the kinematics of the rolling aircraft can be linearized in good accordance to the non-linear model. If no braking occurs, the tire can be reduced to a point follower without slip and without a rolling degree of freedom. Additionally, a rigid attachment from the landing gears to the airframe at a single point is used. Thus, the design model for rolling has been reduced to the 20 most important degrees of freedom.

Basic studies for active suspension control found in the literature are frequently performed using a so-called "quarter-car" or "two-mass"-model in which the vehicle is represented as a mass on top of a single, complete suspension consisting of the shock absorber components, wheel and tire. Such a model, i.e. the aircraft nose landing gear with an equivalent mass, is the final simplification step.

For the passive optimization and for the control design of the semi-active oleo a "bottom-up" strategy is used. Control structure and starting values for the free parameters are selected using the "two-mass"-model, the fine-tuning of the parameters is performed using the design model ("design-by-simulation"), finally the results are evaluated using the evaluation model.

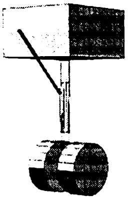


Figure 10: Two-Mass Landing Gear Model

5.3 Passive and Semi-Active Damper

As stated above, suspension requirements, especially the damping value, differ for touch-down and for taxiing. Since the damping force F_{damp} is a function of damping factor d and oleo stroke velocity v_{oleo} ($F_{damp} = d * v^2_{oleo}$) and the oleo velocity at touch-down is relatively high, a low damping factor is required allowing the landing gear to make use of the full oleo stroke. At rolling, however, the stroke velocity is considerably lower, so the low damping value resulting from the landing requirements leads to low damping forces unable to reduce vertical aircraft dynamics like aircraft pitch and fuselage vibrations. As a solution, landing gear can be equipped with mechanical valves to achieve damping factors of a prescribed function of oleo stroke velocity (passive variable oleo - see figure 11) or with electrohydraulic valves which can be controlled by a force law (semi-active oleo - see figure 7). Control input for the semi-active oleo is the aircraft vertical velocity which is obtained by the integration of a measured acceleration signal. A high-pass filter is used to filter out frequencies below the aircraft pitch fre-

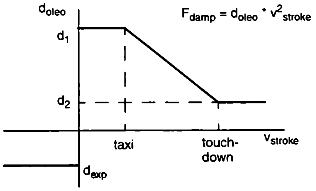


Figure 11: Passive Variable Oleo, Damping Factor Function

quency. The signal is then amplified by a gain P and fed back into the oleo. A semi-active oleo has a lag characteristic which has to be taken into consideration at the control design.

The control structure for the semi-active oleo has been built up in MATRIXx / System Build in the form of block diagrams with the SIMPACK model running in co-simulation mode. After the basic control structure has been established the controller is exported as C-code to be implemented as a SIMPACK User Function leaving the feedback gain as a free parameter to be optimized by SIMPACK - MOPS.

5.4 Optimization

Optimization Cases:

A touch-down case and a rolling case were defined as reference cases for the optimization.

The touch-down was a three-point landing with a vertical velocity of 10 fps (3.05 m/s) at stall speed. Additionally, the oleo was not to reach the limit stroke at a vertical velocity of 12 fps (3.66 m/s) at a forward velocity of 0 m/s (FAR 25.723).

The rolling case was a fast taxiing using a forward speed of 60 m/s. Excitation was a quasi-stochastic runway.

Free parameters:

Free parameters of the optimization were the damping factor for touch-down (d_2) and taxi (d_1), see figure 11, each corresponding to an orifice cross-section, as well as the optimal gain P for the sky-hook damping concept (see figure 8) which is also converted into a commanded orifice cross-section.

Optimization criteria:

The optimization criterion for the landing impact is the vertical load in the landing gears, the criteria for taxiing are the maximum and the RMS of vertical cockpit acceleration

Optimization Procedure:

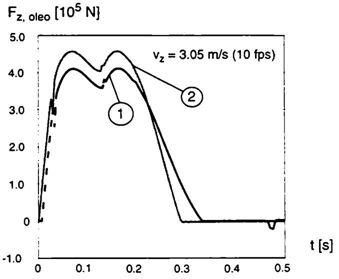
The optimization was performed in several steps. At first good starting values for d_1, d_2, and P had to be determined. To keep the computational effort as small as possible these values were obtained by optimizing a NLG drop test model (see figure 10) in two separate loops, one for landing impact, one for taxi.

The values for d_1, d_2, and P found this way were used as starting points for the overall optimization of the passive oleo and the semi-active oleo, first with the design model, then with the evaluation model.

6. Results

6.1 Passive Landing Gear - Touch Down

In a first step a "two-mass" nose landing gear was subject to a drop-test optimization. The optimal damping coefficient was found to be 65000 $N/(m/s)^2$, with a maximum load of $4.1*10^5$N (see figure 12, curve 1). In a second step, a full aircraft model was used to optimize the nose landing gear oleo. In this case, the optimal damping coefficient was 80000 $N/(m/s)^2$. The maximum loads for this case were found to be $4.4*10^5$N (see figure 12, curve 2). Additionally, a three-point full aircraft landing with the oleo damping value found using the nose landing gear drop test model (65000 $N/(m/s)^2$) led to a maximum load of $4.7*10^5$N, well above the load predicted by the drop test and higher than the result found by optimization of the full aircraft.



1: force in NLG drop test, d_{opt} = 65000 $N/(m/s)^2$

2: A/C three-point landing, d_{opt} = 80000 $N/(m/s)^2$

Figure 12: Results for Landing Impact

These results show that the oleo layout only with a drop test model according to FAR 25 does not necessarily lead to satisfactory results for the full aircraft. A landing gear equipped with an oleo designed only with the help of the drop test model (as would be sufficient to satisfy FAR 25.732) will in this case induce higher loads into the airframe than a landing gear which has been optimized with a full aircraft, further proof for the necessity of full aircraft simulation.

6.2 Optimization for Taxi

Passive Gear:
Rolling simulations with a two-mass NLG model over the stochastic runway were performed to obtain results for the damping coefficients d_l at compression and d_{exp} at expansion. To visualize the basic relation between these two factors a parameter variation over d_l and d_{exp} at a constant velocity of 60 m/s is shown in figure 13, the criteria being the RMS and the maximum value of cockpit acceleration.

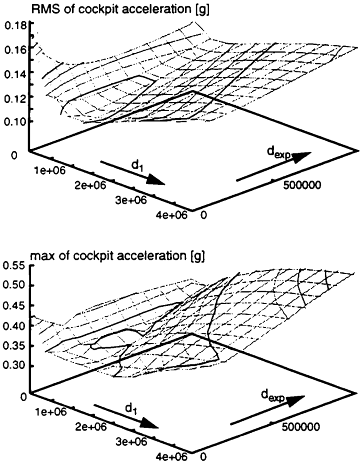


Figure 13: Results for Rolling

The simulations show that the lowest average accelerations (figure 13, top) can be obtained with a relatively high damping coefficient for compression and a much lower coefficient for expansion. The minimum in the plot for maximum acceleration (figure 13, bottom) is not as obvious to find as the one in the RMS-plot but the conclusion is the same - the optimal choice is high damping for compression and low damping for expansion.

A value of $d_l = 1750000$ N/(m/s)^2 was found to be optimal for the complete aircraft optimization. The best value for d_{exp} was found to be 50000 N/(m/s)^2.

Semi-Active Oleo:
In the semi-active gear the force is determined by the orifice cross section between the oleo chambers through which the oil is pressed at compression. This cross section determines the damping factor. The controller commands a damping factor via the feedback of the velocity and a gain P which has been optimized. For a mechanical application the commanded damping factor would have to be recalculated using the equations for oil flow through an orifice to result in the commanded cross-section.

The optimal gain for the test runs was found to be $P = 5.0*10^7$.

Comparison of Results for Rolling:
The reason for introducing an oleo with variable damping was that an oleo optimized for landing performs poorly when the aircraft is rolling at taxi or take-off. This fact is shown by simulations displayed in figure 14: a model of a large transport aircraft rolls over a double-cosine bump similar to that used for the simulations in figure 2. Curve number one is that of the aircraft equipped with a fixed-orifice oleo optimized for the three-point landing. It can be clearly seen that this configuration leads to high vertical accelerations due to the low damping coefficient. A variable passive damping oleo, however, can effectively reduce the vertical oscillations (curve number two). The best result is shown by the semi-active oleo (curve number three). Contrary to the optimized passive oleo it will not only reduce aircraft pitch and heave but is also able to effectively damp elastic modes of the structure and thus reduce airframe oscillation up to a factor of 1.5 to 5, depending on the frequency spectrum of the excitation and the aircraft speed.

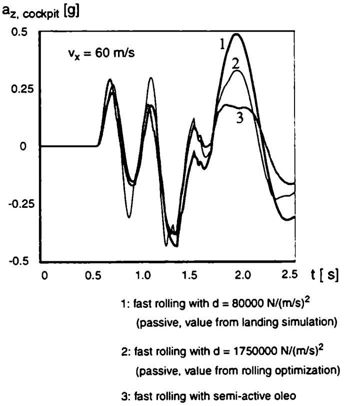


Figure 14: Comparison of Results for Rolling

The advantages of semi-active damping and the dependency on aircraft velocity can be demonstrated by figure 15. The RMS value of cockpit acceleration is plotted for simulation runs of variable aircraft velocity for two aircraft mass configurations. The improvement of the semi-active over the passive suspension increases with aircraft velocity. The concept seems to be more effective for less loaded aircraft due to the fact that the air spring of the heavier aircraft operates on a steeper part of the slope, thus reducing the effective travel of the suspension.

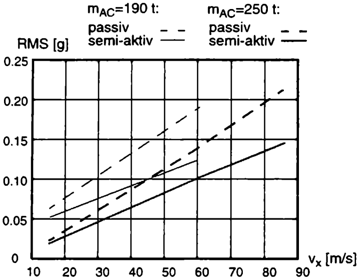


Figure 15: Performance of Semi-Active Oleo

7. Projects

The aircraft / landing gear design is an example of integrated system design in aerospace engineering. DLR is involved in two research projects in this subject. In the first, the German national "Flexible Aircraft" project, DLR plays the role of a link between airframer and landing gear manufacturer who are interested in airframe loads - respectively landing gear loads -and certification. Using the model expertise of the specialists, DLR develops methods and software to help at the exchange of know-how, models and results between the partners (see figure 16). Multibody simulation plays an important role at this project as the central analysis tool from the pre-development phase to calculations for certification, [23].

Another project is the European ELGAR (European Landing Gear Advanced Research) project. The project has the objective to investigate into the possibilities to reduce landing gear weight and to evaluate new concepts in landing gear design, e.g. carbon fibre actuators and semi-active oleos. A number of simulation programs and modeling approaches are compared for their use in aircraft / landing gear simulation and analysis, [24].

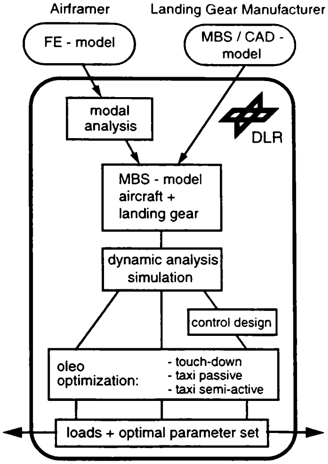


Figure 16: "Flexible Aircraft" Workshare

8. Summary and Future Trends

This study has shown that multibody simulation is a valuable tool in the integrated design process. It allows the integration of modeling expertise from different disciplines, fast system analysis and a fast exchange of results. The necessary tools and methods have been developed at DLR which plays a linking role between aircraft and landing gear manufacturer.

Simulation allows the detection and analysis of parameter interdependencies in a broad range. The engineer can perform test runs on "virtual prototypes" and detect problems at an early design stage, which helps at the reduction of costly full-scale experiments and test flights. Furthermore, new concepts like semi-active damping can be evaluated already in the design phase.

As an example of such a concurrent engineering process an optimization of a passive and a semi-active landing gear has been performed. An aircraft model consisting of a flexible structure supplied by the airframer and a landing gear model from the landing gear manufacturer was the basis of this optimization. It could be shown that design methods using only landing gear drop tests or rigid aircraft models result in aircraft dynamics that are unsatisfactory in respect to loads and comfort. Furthermore, it could be shown by simulation that the damping coefficient optimal for a hard landing impact leads to high vertical aircraft vibrations at taxi. A passive solution can be a velocity dependent damping orifice realized by a mechanical valve.

The best performance, however, can be reached by semi-active landing gears, with low sensor requirements and control structure complexity.

A central point of future work will be the development of centralized databases and the harmonization of the design environments among the partners in the aircraft design.

Much work is also done at the integration of advanced aerodynamics into the analysis of system dynamics and system optimization including loads calculation and flight control design. Finally, the potential of the semi-active landing gear control can be advanced by more sophisticated control structures and the use of semi-active landing gears also for the control of the oleo forces at landing impact.

9. References

[1] S.F.N. Jenkins: *Landing gear design and development*. Proceedings of the Institution of Mechanical Engineers, Vol 203, 1989

[2] Donald W.S. Young: *Aircraft landing gears: the past, present, and future. Aircraft Landing Gear Systems*, Society of Automotive Engineers, pp. 179-196, 1986

[3] Roloff, G.: *Landing Gear Integration on a Supersonic Transport Aircraft*, ICAS-96-4.11.1

[4] Chai, S.T and Mason, W.H.: *Landing Gear Integration in Aircraft Conceptual Design*. Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 1996 (rev. 1997)

[5] Conway, H.G.: *Landing Gear Design*. The Royal Aeronautical Society, 1948

[6] Currey, N.S.: *Aircraft Landing Gear Design: Principles and Practices*. American Institute of Aeronautics and Astronautics, 1988

[7] Joint Aviation Requirements, Chapter 25, Change 13, Joint Aviation Authorities Committee, 1989

[8] Wu, J.K. and Choong, F.N.: *Data and Process Models for Concurrent Engineering of Mechanical Systems*, in: Haug, E.G., "Concurrent Engineering: Tools and Technologies for Mechatronic System Design", Springer, 1993

[9] Ehrlenspiel, K.: *Integrierte Produktentwicklung*, München: Hanser Verlag; 1995

[10] Kortüm, W., Rulka, W. and Spieck, M.: *Simulation of Mechatronic Vehicles with SIMPACK*. MOSIS 1997, Ostrava, Czech Republik 1997

[11] Kortüm, W., Rulka, W. and Eichberger, A.: *Recent Enhancements of SIMPACK and Vehicle Applications*. European Mechanics Colloquium, EUROMECH 320, Prague 1994

[12] Kortüm, W., Schwartz, W., Wentscher, H.: *Optimierung von aktiven Fahrzeugfederungen basierend auf nichtlinearen Mehrkörpermodellen*. Automatisierungstechnik 11/96, Oldenbourg Verlag, München 1996.

[13] H. Wentscher and W. Kortüm: *Multibody Model-Based Multi-Objective Parameter Optimization of Aircraft Landing Gears*. Proceedings of IUTAM-Symposium of Optimization of Mechanical Systems, Stuttgart 1995

[14] Hitch: *Aircraft Ground Dynamics*. Vehicle System Dynamics, Vol 10, 1981, pp. 319-332

[15] P. T. Somm, H. H. Straub and J. R. Kilner: *Adaptive Landing Gear for Improved Taxi Performance*. Boeing Aerospace Company,1977, AFFDL-TR-77-119

[16] W.E. Howell, J.R. McGehee, R.H. Daugherty, W.A. Vogler: *F-106 airplane active control landing gear drop test performance*. Landing Gear Design Loads (CP 484), AGARD, 1990

[17] Tyrone Catt, David Cowling and Alan Shepherd: *Active landing gear control for improved ride quality during ground roll*. Smart Structures for Aircraft and Spacecraft (AGARD CP 531), Stirling Dynamics Ltd, Bristol, 1993

[18] Wentscher, H., Kortüm, W., Krüger, W. R.: *Fuselage Vibration Control Using Semi-Active Nose Gear*. AGARD Report 800: „The Design, Qualification and Maintenance of Vibration-Free Landing Gear", 1996.

[19] D. Karnopp: *Active Damping in Road Vehicle Suspension Systems*. Vehicle System Dynamics, 12(6), Swets & Zeitlinger, Lisse, 1983

[20] H. Wentscher: *Design and Analysis of Semi-Active Landing Gears for Transport Aircraft*. DLR Forschungsbericht 96-11, 1996

[21] International Organization for Standardization: Guide for the Evaluation of Human Exposure to Whole Body Vibration. ISO 2631-1974(E), July 1974

[22] Meirovitch, L. and Kwak, M.K.: *On the Modeling of Flexible Multi-Body Systems by the Rayleigh-Ritz-Method*, 1990, Long Beach

[23] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie: *Statusseminar Leitkonzept MEGALINER, Berichte aus dem Luftfahrtforschungsprogramm*, Hamburg, Juni 1997

[24] Krüger, W. R., et al.: *Landing Gear Dynamics: Simulation and Control*. in: Vehicle System Dynamics, Swets & Zeitlinger, Lisse, 1997

Dynamic Simulation of an Articulated Off-Road Vehicle

Kazuya Yoshida, [*] Tomoko Shiwa [†]
and Mitsunori Oda [‡]
Tohoku University, Sendai, JAPAN

In the space robotics lab, Tohoku University, we have developed a laboratory test-bed of a planetary exploration rover, which has an articulated chassis with both active and passive wheels. The test-bed has shown a good locomotion capability on natural rough terrains. In this paper, we focus the tire mechanics based on the Gim and Nikravesh model. The kinematics and dynamics of the articulated rover is studied for practical computer simulation. The dynamic simulation with the presented tire model shows a good agreement with the experimental result for the winding motion on a flat floor.

Introduction

PLANETARY exploration rovers have recently received much attention from the science community as a device to investigate geology and explore evidence of life on remote plants. Following to the great success of the JPL/NASA PathFinder mission, Japanese space agencies including ISAS (Institute of Space and Astronautical Science) and NASDA (National Space Development Agency of Japan) are organizing potential roving and exploration missions on Moon. These missions are expected to be supported by science and engineering teams formed in various universities. In spite of the extensive research activities on mobile robots in many Japanese universities, only few are seriously focusing on planetary exploration. In Tohoku University, we are initiating an intensive research effort on planetary rovers, keeping close ties with those space agencies. Our primary goal is to obtain a fundamental knowledge on the mechanics of locomotion and navigation of a vehicle moving on a natural rough terrain, such as rocky or sandy surface. Toward this goal, we have developed a laboratory test-bed which has an articulated chassis adaptive to rough terrains. Using this test-bed, we observe the dynamic behavior of vehicle, especially including the slip and traction of tires, so that we can built up a useful model for navigation and control.

In this paper, the dynamic behavior of the articulated off-road vehicle is simulated with special attention to a slip model of tires and articulated dynamics of the chassis. The vehicle test-bed has a design like Bekker type chassis,[1] but mounts both traction (active) and odometer (passive) wheels, so that the difference in the rotations of those wheels gives us slip measurement. The articulated bodies are connected by a passive linkage whose joint angles are measured.

For the dynamic simulation, a simple but useful tire model is investigated, paying attention to slip and traction characteristics. A general model of multibody dynamics, which is originally developed for space free-flying robots or structures, is applied to simulate the articulated chassis dynamics.

Using the laboratory test-bed, the slip rate and slip angle of tires are measured for various terrain conditions. The developed simulation is applied to predict the motion of the vehicle based on the measurement of the tire parameters, then compared with the actual motion.

Design of a Rover Test-Bed

A lot of rover designs are developed and tested for tough and reliable mobility on natural rough terrains. The Rocker-Bogie system used in the *Sojourner* rover is one of the most successful design as is proven in the Mars Path Finder mission 1997. On the other hand, Russians have developed *Marskhod* that shows very tough mobility and, looking back the history, JPL also has studied articulated chassis from SLRV project[2] in 1960's to *Robby* test-bed. Of course, there many other designs in the world.

Among them, we chose an articulated chassis with 6 wheels, since our goal of research is not a challenge of new designs, but a rapid-prototyping of a robust and durable platform. Eventually an articulated chassis is relatively simple to design and built by university students, yet we can get robust enough locomotion capability on rough terrains. An articulated type chassis has been extensively studied by Bekker,[1] therefore this type of chassis can be called *Bekker type*. But unlike original Bekker type, our test-bed has the design with four traction (active) wheels and two passive wheels for odometory without slip, and passive linkage with joint angle measurement instead of a flexible coupling.

Figure 1 shows a kinematic configuration of our

[*] Associate Professor, The Space Robotics Lab., Dept. of Aeronautics and Space Engineering, Aoba 01, 980-8579, AIAA member.

[†] Graduate student

[‡] Currently works for Ishikawajima Heavy Industry, Ltd.

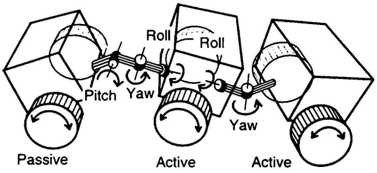


Fig. 1 Configuration of our rover test-bed

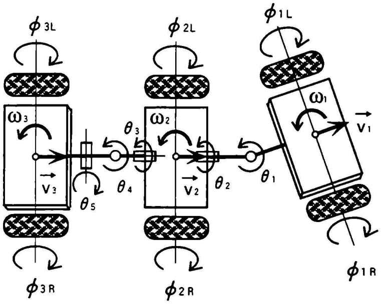


Fig. 2 Kinematic model

rover test-bed. Six wheels are mounted on 3 articulated bodies. Each body (we call *segment*) has 2 wheels. The segments are connected by 2 DOF passive joints, yaw and roll, between the first and second segments. And by 3 DOF passive joints, yaw, pitch, and roll, between the second and third segments. Roll joints are free, but yaw and pitch joints have compliance. Top 4 wheels on the first and second segments are active and independently driven by DC motors, whereas last 2 wheels on the third segment are passive and used as tracer wheels, or odometers, for dead-reckoning measurement and slip estimation.

The rotation angles of all wheels and joints are measured by potentiometers.

Rover Test-Run

The specification of the rover test-bed is listed in Table . Figure 3 shows a picture of rover test-run on a snowy surface. We tried several bumpy surfaces, and it is confirmed that the test-bed has good adaptivity and locomotion capability on rough terrains.

The dead-reckoning accuracy is evaluated by comparison between the physical location of each segments and the computed one from odometory and the kinematic model. The error in the tangential direction is less than 1 percent in case of the locomotion on a flat floor.[4]

It is observed that each active tire has non-negligible slip during the motion, and this fact suggests that it is necessary to understand the nature of the slip then to develop a controller to minimize the slip.

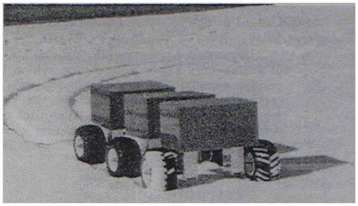


Fig. 3 Test-run on a snowy surface

Table 1 Rover Specifications

Dimensions	[mm]
Length × Width × Height	90 × 500 × 330
Wheel Radius	170
Total Weight	16.4 [kg]
DC Motor	0.3 [kg] × 4
Battery	0.7 [kg] × 2

Kinematic Model

Figure and Table 2 show kinematic model and notation of our test-bed.

We here assume that a pair of tracer wheels on the third segment have no slip. Then the linear and angular velocities of the third segment are obtained from the measurement of their rotation (ϕ_{3R}, ϕ_{3L}), such as:

$$\left\{ \begin{array}{l} \vec{w}_3 = d\,(\dot{\phi}_{3R} - \dot{\phi}_{3L})\,\vec{k}\;/\;2r \\ \vec{v}_3 = d\,(\dot{\phi}_{3R} + \dot{\phi}_{3L})\,\vec{i}\;/\;2 \end{array} \right. \tag{1}$$

Using the kinematic relationship among the segments and joint angle measurements, we can calculate the linear and angular velocities of the first and second segments:

$$\left\{ \begin{array}{l} \vec{w}_2'' = \vec{w}_3 + A_5\vec{k}\,\dot{\theta}_5 \\ \vec{v}_2'' = \vec{v}_3 + \vec{w}_3 \times \vec{b}_6 + \vec{w}_2'' \times A_5\vec{a}_5 \end{array} \right. \tag{2}$$

$$\left\{ \begin{array}{l} \vec{w}_2' = \vec{w}_2'' + A_4\vec{k}\,\dot{\theta}_4 \\ \vec{v}_2' = \vec{v}_2'' + \vec{w}_2'' \times A_5\vec{b}_5 + \vec{w}_2' \times A_4\vec{a}_4 \end{array} \right. \tag{3}$$

$$\begin{cases} \vec{w}_2 = \vec{w}_2' + A_3\vec{k}\,\dot{\theta}_3 \\ \vec{v}_2 = \vec{v}_2' + \vec{w}_2' \times A_4\vec{b}_4 + \vec{w}_2 \times A_3\vec{a}_3 \end{cases} \tag{4}$$

$$\begin{cases} \vec{w}_1' = \vec{w}_2 + A_2\vec{k}\,\dot{\theta}_2 \\ \vec{v}_1' = \vec{v}_2 + \vec{w}_2 \times A_3\vec{b}_3 + \vec{w}_1' \times A_2\vec{a}_2 \end{cases} \tag{5}$$

$$\begin{cases} \vec{w}_1 = \vec{w}_1' + A_1\vec{k}\,\dot{\theta}_1 \\ \vec{v}_1 = \vec{v}_1' + \vec{w}_1' \times A_2\vec{b}_2 + \vec{w}_1 \times A_1\vec{a}_1 \end{cases} \tag{6}$$

where, $\vec{v}$ and $\vec{w}$ with prime and double prime are the inertial velocities of the connecting links between the segments.

By solving above velocity equations for each wheel, we can get the nominal rotation angles of the active wheels in the ideal situation without any slip.

$$\begin{cases} \dot{\phi}_{2Rnominal} = (-r\vec{w}_2 - \vec{j} \times \vec{v}_2)_z \;/\; d \\ \dot{\phi}_{2Lnominal} = (r\vec{w}_2 - \vec{j} \times \vec{v}_2)_z \;/\; d \end{cases} \tag{7}$$

$$\begin{cases} \dot{\phi}_{1Rnominal} = (-r\vec{w}_1 - \vec{j} \times \vec{v}_1)_z \;/\; d \\ \dot{\phi}_{1Lnominal} = (r\vec{w}_1 - \vec{j} \times \vec{v}_1)_z \;/\; d \end{cases} \tag{8}$$

We measure that the rotation angles of the active wheels, $(\phi_{1R}, \phi_{1L})_{actual}$ and $(\phi_{2R}, \phi_{2L})_{actual}$, which eventually include slippage. The slippage is then evaluated by the comparison of the nominal and the actual wheel rotations.

$$Slippage = \dot{\phi}_{nominal} - \dot{\phi}_{actual} \tag{9}$$

Table 2 Kinematic parameters

d	:	wheel radius
r	:	length from center of segment to wheel
$\vec{a}_i, \vec{b}_i$	:	vectors for connecting links
A_i	:	coordinate transformation matrix
	:	corresponding to joint angle θ_i
$\vec{i}, \vec{j}, \vec{k}$	:	principal axes of each body-fixed
	:	coordinate frame

Tire Mechanics

Modeling

In this paper, we develop the model of the tire mechanics based on the paper by Gim and Nikravesh.[3]

A rubber tire is modeled a visco-elastic element to connect the chassis and the ground, with stiffness C and damping D, as shown in Figure 4 and 5. A coordinate frame is defined in a way that the origin is located at the center of the contact region, x axis coincides a rolling tangent direction, y a rolling normal direction, and z a vertical direction.

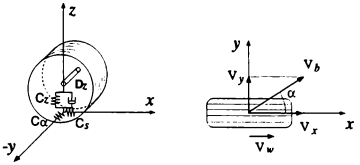


Fig. 4 Tire model with the definition of coordinate system

Slip Rate

The slip rate is defined with the tire velocity v_x, v_y, which are equivalent to the chassis velocity, and the circumference velocity v_w in the following way:

$$S = \begin{cases} (v_x - v_w) \;/\; v_x & (\, v_x \;>\; v_w \,) \\ (v_x - v_w) \;/\; v_w & (\, v_x \;<\; v_w \,) \end{cases} \tag{10}$$

$$S_\alpha = \begin{cases} |\tan\alpha| & (v_x > v_w) \\ (1 - |S|)\,|\tan\alpha| & (v_x < v_w) \end{cases} \tag{11}$$

$$\tan\alpha = v_y / v_x$$

Here $v_x > v_w$ indicates the acceleration of the vehicle, and $v_x < v_w$ deceleration.

Grip and Slide

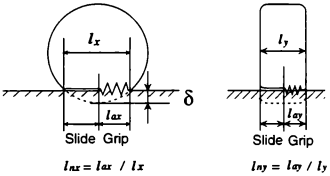


Fig. 5 Tire deformation model

A rubber tire deforms in z direction by δ to support the vertical load, and contacts with the ground in the region of $l_x \times l_y$ in x and y directions respectively. However, all the region does not necessarily grip the ground firmly but only some part from the contact front grips and rest has frictional sliding contact, as shown in Figure 5. The critical point to loose the grip is determined the balance of the elastic force due to the longitudinal deformation of rubber tire along the contact arc, and the frictional force of the arc. The grip ratios l_nx, l_ny are defined with the contact arc l and the grip arc l_a.

$$l_n \;=\; l_a \;/\; l \tag{12}$$

Tire Force

The tire force along x and y directions are then modeled in the following way:

$$F_x = \begin{cases} C_s|S|l_{nx}^2 + \mu_x F_z(1 - 3l_{nx}^2 + 2l_{nx}^3) & \\ \qquad\qquad\qquad\qquad (\ |S| \ < \ S_c \ ) & \\ \mu_x F_z \qquad\qquad\qquad (\ |S| \ > \ S_c \ ) & \end{cases} \qquad (13)$$

$$F_y = \begin{cases} C_\alpha|S_\alpha|l_{ny}^2 + \mu_y F_z(1 - 3l_{ny}^2 + 2l_{ny}^3) & \\ \qquad\qquad\qquad\qquad (\ |S_\alpha| \ < \ S_{\alpha c} \ ) & \\ \mu_y F_z \qquad\qquad\qquad (\ |S_\alpha| \ > \ S_{\alpha c} \ ) & \end{cases} \qquad (14)$$

Here, C_s, C_α are tire stiffness in the grip region, and μ_x, μ_y are friction coefficient in sliding region. The values

$$S_c = 3\mu_x F_z \ / \ C_s \qquad (15)$$

$$S_{\alpha c} = 3\mu_y F_z \ / \ C_\alpha \qquad (16)$$

are critical slip rate. If a tire has a larger magnitude of slip rate than this critical value, all the contact region must be sliding.

On the other hand, the vertical force in z direction is expressed with stiffness and damping C_z, D_z in the following way:

$$F_z = C_z\delta - D_z\dot{\delta} \qquad (17)$$

Articulation Dynamics

For the modeling and simulation of the dynamics of an articulated system, the first author of this paper has a strong background since he was originally motivated by the analysis of a free-flying space robot. A general model for arbitrary articulated systems with plural branches and plural contact points [1] is developed in the form of MATLAB toolbox, and applied to many robotic systems and space systems.[5,6]

For the rover model presented here, segment 3 is treated as the reference body and a main trunk continues to segment 1. On each segment a set of tires are mounted as branches and, at the end tip of each branch, the tire forces F_x, F_y, F_z modeled by equations (13), (14), (17) are applied as external forces.

Simulation

Dynamic simulation of a flat surface locomotion is carried out using roughly estimated tire parameters as listed in Table 3. As an input of the simulation, the circumference velocity of each active wheel as:

$$v_{w1R} = v_{w2R} = 0.05 + 0.015\sin(\pi/25 \cdot t) \qquad (18)$$

$$v_{w1L} = v_{w2L} = 0.05 - 0.015\sin(\pi/25 \cdot t) \qquad (19)$$

(the velocities are in [m/s], time t in [s].)

so that the rover has a winding motion.

Table 3 Tire Parameters used in the simulation

friction coefficient	μ_x	0.4
	μ_y	0.5
tire stiffness	C_s[N/m]	100
	C_α[N/m]	50
	C_z[N/m]	1000
tire damping	D_z[Ns/m]	72.2

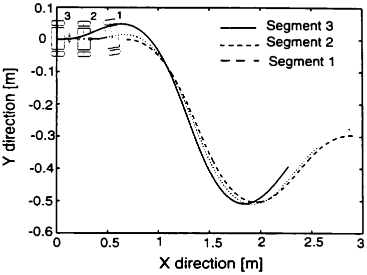


Fig. 6 Motion trace: Simulation

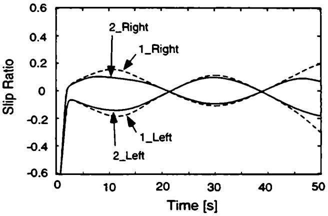


Fig. 7 lip ratio profile: Simulation

Figure 6 is the resultant motion trace of the rover, and Figure 7 is the slip rate during the motion. From these figures, it is observed that, during a steering motion, the wheels outer side of the curve have a negative slip and the wheels inner side of the curve have a positive slip. This is a very reasonable result for differential steering.

Experiments

An experiment to be compared with the above simulation is carried out on a flat linoleum floor with the same tire velocity command as equation (18) and (19). Figure 8 is the dead-reckoning motion trace of the rover, and Figure 9 is the slip rate during the motion.

Comparing the Figure 8 with 6, the traveling distance and the shape of the curve show remarkable

[1] However, a requirement is imposed that the system has topological tree configuration and each contact point is located at the end tip of a branch.

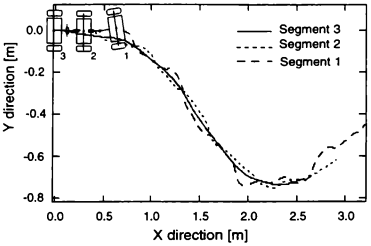


Fig. 8 Motion trace: Experiment

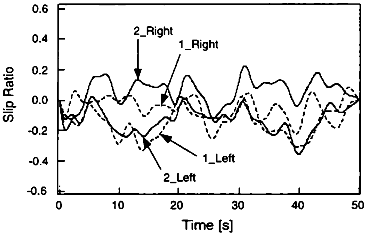


Fig. 9 Slip ratio profile: Experiment

agreement between the simulation and experiment. We however think that further improvement of the tire parameter C_α will be possible to get more precise agreement in the rolling normal direction.

Comparing the Figure 9 with 7, the same trend of inner-positive and outer-negative slip on curve is clearly observed. Although the experimental data is somewhat noisy, the magnitude of the slip ratio shows a good agreement with the simulation.

Conclusions and Discussions

In this research, we have developed a laboratory test-bed of a planetary rover, which has an articulated chassis with both active and passive wheels. The test-bed has a good locomotion capability on natural rough terrains.

In this particular paper, we focus the kinematic model, tire model and dynamic model of the rover. The dynamic simulation using the presented tire model shows a good agreement with the experimental result for the winding motion on a flat floor. We regard it as a very first step for further comprehension of the locomotion mechanics, and we know that there are still many open questions. Some of them are listed below:

- How can we find a better estimation of the tire parameters? - We need to establish an easy but effective method to identify them. The method can hopefully work to improve the estimation by an on-line manner.

- What about natural terrains, rather than a flat floor? - What we should do first is just try many experiments on various terrains.

- Is the Gim and Nikravesh model of tire mechanics good for solid, non-rubber tires? - From a planetary rover point of view, what we really need is a good model for a metal (aluminium) tire on a sandy or rocky terrain.

- How do we control the tire velocity or torque to avoid significant slip or stack? - This is a final goal of our research.

References

[1] M. G. Bekker: *Introduction to Terrain-Vehicle Systems*, The University of Michigan Press, 1969.

[2] J. Burke: "Missions, Technologies and Design of Planetary Mobile Vehicles," in *Int. Sym. on Planetary Mobile Vehicles*, Toulouse, 1992.

[3] G. Gim and P. E. Nikravesh: "An Analytical Model of Pneumatic Tyres for Vehicle Dynamic Simulations. Part 1: Pure Slips," in *Int. J. of Vehicle Design*, Vol.11 No,6, pp.589–618.

[4] T. Shiwa and K. Yoshida: "Navigation Data Acquisition with an Experimental Test-Bed of Planetary Exploration Rover," in *Proc. of the Int. Conf. on Field and Service Robotics*, pp.109–114, Canverra, Australia, 1997. (to appear in a post conference book)

[5] K. Fujishima, T. Hiraoka, and K. Yoshida: "Dynamic Simulation of Free-Flying Space Robots for Service and Exploration," in *Proc. of the Int. Conf. on Field and Service Robotics*, pp.456–462, Canverra, Australia, 1997. (to appear in a post conference book)

[6] K. Yoshida and D. N. Nenchev: "A general formulation of under-actuated manipulator systems," in *Preprints of the 8th Int. Symp. on Robotics Research*, Japan, 1997.

AIAA-98-4363

PREDICTIVE REFERENCE MODEL BASED CONTROL OF THE SIMONA FLIGHT SIMULATOR MOTION SYSTEM.

S.H. Koekebakker* Ph. Piatkiewitz* S.K. Advani° L.J.J. de Nijs°

International Centre for Research in Simulation, Motion and Navigation Technologies SIMONA
Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands, s.h.koekebakker@wbmt.tudelft.nl

Keywords: *Predictive control, Reference model based control, Flight simulator motion*

Abstract

In this paper, a synergistic approach towards the control of a flight simulator motion system, with the goal of realizing effectively unnoticeable time delays, is presented. Traditionally, the vehicle model and motion drive (washout) filter modules calculate the desired motion trajectory, which the controlled motion system then attempts to realize as fast as physically possible. Due to the physical properties of the system however, this inevitably results in lags or delays of the actual versus the desired motion. A strategy in which the motion system is controlled, such that its dynamics are turned into the dynamics of the vehicle to be simulated, would be one way to overcome this problem. The modular structure, however, would be lost through this approach. A compromise between these two approaches is proposed here: A reference model comes up with predictions 30-50 ms ahead of the desired vehicle motion. Smooth interpolation prevents any high frequency dynamics distortion of multiple-sampled systems. Finally, the smooth prediction can be used in a feed-forward setting of the model-based motion system controller. The new approach is validated by a benchmark set of simulator critical maneuvers. Experiments with the SIMONA motion system, with a 4000 kg payload, demonstrated remarkable improvement.

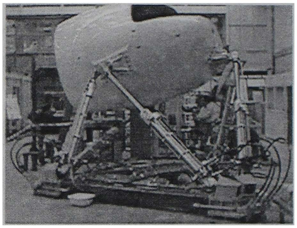


Figure 1: Simona research simulator motion system

Introduction

In order to increase the effectiveness of simulators in lieu of in-flight training and research activities, the SIMONA institute conducts research into flight simulation techniques. A central element of this institute will be the SIMONA Research Simulator (SRS) of which the motion system construction has been completed.

In flight simulation, the pilot relies on the perception of self-motion through several stimuli, and uses this perception to exercise control over the aircraft. Stimuli cueing systems include the visual system, the motion system, the audio system, control loading and the aircraft instruments stimulation. In the control task, the pilot uses his visual perception to make a good estimation of the aircraft's attitude and velocity. The approximate frequency response of visual perception can be modelled as a first order low-pass filter with a break frequency

*Mechanical Engineering Systems and Control Group
°Aerospace Engineering Control and Simulation Group
S.K. Advani is director of SIMONA, member AIAA

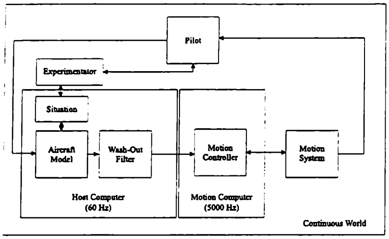


Figure 2: Block diagram representation of motion simulation

of 0.1 Hz[9], which is rather slow. Motion is however also perceived by the pilot's vestibular and tactile sensors, which are sensitive to specific forces and angular accelerations. These signals are processed rapidly by the central nervous system and, therefore, give the pilot lead information. With this lead information, the pilot can react more quickly to changes in the vehicle motion state. This information is most important at the higher frequencies (above the bandwidth of visual perception). These high-frequency motions are, in simulation, often called onset-cues.

Because of the high-frequency nature of these onsets, it is important that the motion system has sufficient bandwidth. Moreover, it is important that the time-delay in the motion simulation is kept as small as possible, since an onset is also time-critical. If the onset is simulated noticeably late, simulator training quality will be decreased considerably. Time delays, furthermore, lower the pilot-vehicle crossover frequency and may require the pilot to adapt by applying lead compensation. The reduction of the adaptation required is precisely one of the goals of future flight training[1]. Therefore, a primary objective of a flight simulator motion system is to provide the pilot in the simulator with appropriate representations of, what one could call, the "generalized forces" i.e. both translational and rotational accelerations and gravity. This has to be attained by driving the simulator with six parallel hydraulic servo actuators. With the SRS, an important basis for high performance motion has been attained by construction design for control including low-mass, low centre-of-gravity, and a high-stiffness structure[2]. Its potential can only be fully exploited by advanced controller design techniques.

With the application of hardware components, like fast multi-processor Digital Signal Processor (DSP) boards, and software which performs automatic DSP-code generation from higher level programs e.g. Matlab/Simulink, it is possible to use highly structured complex control strategies which take model knowledge into account.

Problem Description

The objective of predictive reference model based feed-forward control is to reduce the aforementioned latencies to effectively zero. This is achieved by using knowledge of the vehicle to be simulated, known as the reference model, to guide the simulator by feed-forward and feedback control, which also accounts for the motion system dynamics.

Providing the motion controller solely with the desired system's output results in latencies, since pure feedback has limited bandwidth. This observation led to the key idea that predictive knowledge from the fully available simulation model should be obtained and used in an appropriate way.

To obtain a solution that can be implemented on and properly integrated with the available subsystems, namely the host and motion computer, the problem was split into the following:

- reference model prediction
- reference model based feed forward control design
- systems interconnection

These problems will be treated separately in the subsequent sections of this paper.

Reference model based predictors

The simulation model can be used as a reference signal generator. The only uncertain factor is the pilot. However, due to the relatively slow response by the human operator, it is quite easy to predict with reasonable accuracy the future accelerations over a short period of time (30-50 ms). The major problem is due to the complexity of the vehicle model, making it difficult to calculate future values within the real-time environment. Approximate models can however be used, and several methods have been studied and evaluated[4].

Approaches

The model approximation methods ranged from the very general method of series expansion (with little knowledge of the simulation model), to the fast-time modelling approach, in which the structure of the simulation model is taken explicitly into account.

- **Taylor series expansion.**
  With the most recent data points of the simulated signals, a polynomial is fitted. Extrapolation of this polynomial results in a prediction of the future values of the signal.
- **Identified linear model predictor.**
  A linear state-space model is identified from the non-linear time-variant vehicle model. With the identified model and its state, a future prediction can be made.
- **Self tuning predictor.**
  The previous model parameters can be updated by an adaptation using a stochastic model and correlation methods.
- **Fast time modelling (FTM).**
  The simplified model of the equations of motion of the aircraft can be updated by the coefficients calculated by the original (and more complex) simulation model. In the construction of the predicted motion, these coefficients are assumed to be constant.

The advantage of the first three methods is that they can be used more independent of the specific vehicle to be simulated. However, the expansion method and self tuning predictor appeared to suffer heavily from high frequent distortion like a turbulence model. One constant linear approximation model was not able to come up with high quality predictions at all operation points. The FTM method proved to be best suited for prediction over limited time horizons. It revealed constant performance under different test conditions.

The parameters transfered to a linear model of the equations of motion of the aircraft are:

- Aerodynamic coefficients,
- Thrust,
- Mass parameters,
- Aircraft state vector.

With these parameters together with the wash out filter parameters (which do not have much effect on the short horizon as they merely act as high pass filters) the required acceleration of the simulator can be obtained

The total of forces and moments acting on the aircraft is a summation of those generated by the engine model, aerodynamic model, landing gear model and turbulence model. As forces immediately result in accelerations considering the equations of motion, the prediction horizon should be obtained by the ability to predict the future forces smoothly over some time frame. As thrust, aileron, elevation and rudder steering and landing gear model have reasonable time constants, this can be done with respect to the engine, aerodynamic and landing gear forces. The turbulence related forces vary considerably faster but as the dependence on the pilots actions can be assumed constant over short horizons, the stochastic nature can be simulated in advance to deliver future predictions.

Reference model based control

The most relevant system dynamics[7] and the multiple control approach[8] have been reported on earlier and the main results will shortly be presented first. Then it is discussed what options for a reference model based controller can be chosen and how this can be integrated with the over all control structure.

Motion system characteristics

The basic dynamics of a hydraulically driven mechanical system such as the flight simulator motion system of the SRS can be represented by the block scheme given in Fig. 3. It consists of equations which describe the hydraulics on the left side and the mechanics on the right side. The hydraulic elements of the equation are

$$\dot{\bar{p}} = C(V\bar{i} - L\bar{p} - A\dot{\bar{q}}). \qquad (1)$$

This vector equation is a flow balance. The flows given by, $\bar{\phi}$, result from the input signal to the valve, $\bar{i}$. The

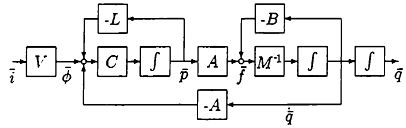


Figure 3: Basic structure hydraulically driven motion system.

valves have a dynamic transfer function matrix V. The actuators move with velocities, $\dot{\bar{q}}$. The actuator chambers have to be filled with oil and the flow necessary to fill these volumes equal the actuator areas A times the velocities, $\dot{\bar{q}}$. Finally the flow balance contains a term which stands for the leakage oil which equal the relative leakages L times $\bar{p}$. The unbalance tends to raise the pressures through the oil stiffnes, C.

On the right side (see Fig. 3) the mechanics are modelled.

$$M\ddot{\bar{q}} = A\bar{p} - B\dot{\bar{q}} \tag{2}$$

The net resulting forces, $\bar{f}$, will accelerate the platform through the inverse mass matrix, M^{-1}. This matrix depends on the platform pose and therefore is continously varying as the simulator moves. The net forces consist of the actuator forces, $\bar{f}_a = A\bar{p}$, and the viscous friction, $B\dot{\bar{q}}$. Gravity, coriolis and centripetal forces are not shown here for the sake of simplicity.

An important aspect with respect to the reference model based control is that there is no direct response of the acceleration (or force generation) resulting from the control inputs, $\bar{i}$. This is due to the presence of the valve dynamics and the finite oil stiffness in the column.

Multiple level control

Conventionally, flight simulator motion systems are controlled by local feedback of the individual actuator positions. The response is then damped by pressure feedback. The reference actuator positions are in fact calculated from the desired simulator pose. The acceleration reference can also provide for some feed forward.

With the SRS, another approach has been taken: With the application of model-based control, the desired motion outputs (namely, the specific forces and angular accelerations) can be driven more directly. Therefore the design of a feed forward signal becomes more straight forward. Locally, the hydraulic actuators can be turned into smooth force generators by introducing a pressure feedback term. Further a velocity compensation cancels the feedback path, $A\dot{\bar{q}}$, in Fig. 3. Thereby decoupling the hydraulics from the mechanics. This approach was already succesfully applied earlier[6].

The inner loops form the lower level in the control structure. This structure is given in Fig. 4. It consists of the following:

Level 1. Local hydraulic pressure control loops[6,12,13].

Level 2. Multivariable feedback linearization[7,14].

Level 3. Outer loop position stabilization[11].

Level 4. Reference model based control[10, 15].

The required pressures, P_d are calculated by a model of the mechanics. Given the required accelerations, this model (expressed in platform coordinates) can then calculate the required actuator forces. The platform coordinates, X, have to be calculated from the actuator positions, $\bar{q}$. The required accelerations consist of (a) the desired acceleration, calculated by the reference model based feed forward to be designed, and (b) the corrective acceleration of the outer loop position feedback, which has to prevent the motion system actuators from drifting to the end of their strokes.

Reference model based feed forward

First, the requirements for the reference model based control task will be given. Then, the different approaches will be outlined.

Requirements In the ideal case, the multiple-level controller would result in a transfer function from the desired platform accelerations, $\ddot{X}_d$ to actual accelerations $\ddot{X}$ equal to a fast first-order response, determined by the time constant of the pressure feedback.

In practice, however, several other aspects lead to deficiencies in the system. First of all, the velocity compensation is not perfect and, as a result, the pressure feed-

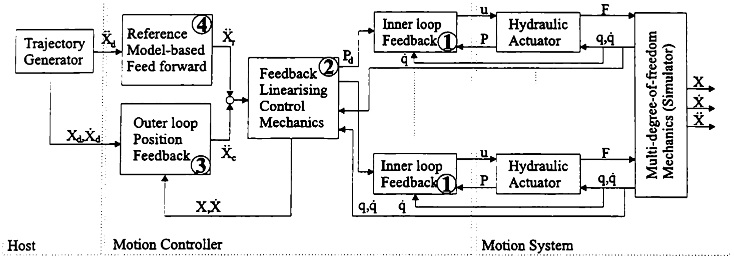


Figure 4: Multi level control structure of the Simona Motion System

back gains cannot be increased to arbitrarily high values. Nonetheless, the hydraulic actuators can be turned into 'force generators' with a bandwidth of about 2-3 times the lowest natural frequency of the rigid-mass system with finite oil spring stiffness. In the case of the SRS, this natural frequency, with a design load of 4000 kg, is ≈ 5-7 Hz. Finally, the limited bandwidth of the servo valves accounts for an additinal latency of about 10 ms. Furthermore, the reference accelerations should not have a frequency content higher than 20-30 Hz: Undesirable deformations of the simulator result in parasatic resonances slightly above this frequency range. The pilot's visual system is highly sensitive to vibrations of the visual display system optics.

Approach A study of literature resulted in three options for a trajectory tracking control approach.

- *Servo Control.* If little use can be made of knowledge of a reference model or of the system to be controlled, the servo control of Desoer[5] still results in the robust asymptotic tracking of a reference signal. Transient response quality, which is considerably relevant in motion simulation where onsets play an important role, cannot be guaranteed however. The hydraulically driven motion system has tracking quality w.r.t. step signals of the position if position feedback is applied due to the physical structure given in Fig. 3.

- *Model Matching.* Vehicle simulation can also be seen as trying to match the dynamics of the 'washed out' vehicle model by the dynamics of the simulator motion system. Model matching[16] can be applied with feed-forward, feed back, or both.

- *Preview Control.* With preview control[15], predicted references can be used in a controller in order to yield limited phase lags for systems not having stable inverses.

The scheme given in Fig. 4 results in tracking for the long term of the platform pose which is stabilized by the outer loop position feedback. Further, as already pointed out, the inner loop feedback and feedback linearising control result in first order responses of the system, $G(s)$, from desired to actual accelerations nominally. This is given by

$$G_{nom}(s) = \frac{1}{\tau s + 1} I, \tag{3}$$

where the time constant τ is determined by one over the product of pressure feedback and the oil stiffness times valve gain, CV. The oil stiffness, C, varies slightly with the position of the actuator[12]. The valve gain also shows some non-linearity.

The immediate unit (step) response of a system can be obtained by precompensation with inverse system. Because the system $G(s)$ is strictly proper, the inverse is not a proper stable system. Two approaches can be taken to overcome this problem.

- First, knowledge of the reference system can be used to determine the derivative acceleration, the 'jerk' signal $d/dt(\ddot{X}_d)$. If

$$\ddot{X}_r = \tau d/dt(\ddot{X}_d) + \ddot{X}_d \quad (4)$$

is used as feed forward, both the phase lag and high frequent amplitude decrease are compensated for. As high frequency components tend to be amplified in this way, one needs to be sure that the reference signal does not contain high frequent components which could excite the parasatic resonances of the motion system. Moreover, as τ is not even exactly known and is not constant, the compensation can be wrong in the mid-frequency area i.e. the bandwidth area of the motion system of 10-20 Hz.

- Alternatively, using a prediction of the future acceleration over a horizon equal to the time constant of the controlled motion system can be used to compensate for the phase lag over the relevant frequency area *without* unecessarily amplifying the high frequency area. The reference to the feedback linearising controller would then be

  $$\ddot{X}_r(t) = \ddot{X}_d(t + \tau). \quad (5)$$

  Also with this approach high frequency reference signal will not be compensated for properly, but at least will not be amplified.

In either case, the connection of the host to the motion computer has to be such that the reference signal does not have a significant high frequency content. This is especially relevant since the host is operating at a relative low sample rate w.r.t. the motion computer. Caution has to be taken to prevent aliasing effects from mapping into this area. This will be discussed next.

Host/Motion computer connection

The host computer generates the simulator feasible trajectories on a sample frequency which is usually set at 60 Hz. In earlier days even 30 Hz was seen to be used. In near future, one is likely to be able to set this frequency at 120 Hz.
According to Shannon's theorem, this means that reference signal frequencies higher than half the sample frequency, $> .5f_n$, can not be discriminated from the area $0 \ldots .5f_n$. In practice, aircraft model time constants should be well below this frequency. This alone motivates a higher sampling frequency as also flexible effects of large aircraft or rotorcraft need to be taken into account in simulation models in the future.

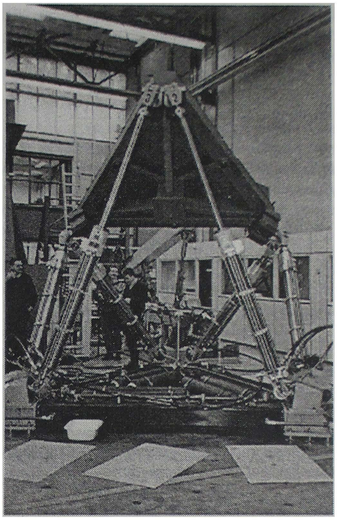


Figure 5: The motion system of the SRS with an temporary dummy platform load of 4000 kg in the central workshop of mechanical engineering.

Extensive literature[3] towards signal processing and filtering in the area of multi-rate signal processing exist. Since the motion system control computer requires (smooth) signals at 1-5 kHz, the lower-frequency sampled signals of the host computer will have to be interpolated. Ideally, if the signal $x(k)$ to be interpolated would be known from time, t, in the interval $-\infty$ to ∞, filtering with the well known anti-causal filter $h(t)$,

$$
\begin{aligned}
x_c(t) &= \sum_{k=-\infty}^{\infty} h(t - k/f_n)x(k) \\
&= \sum_{k=-\infty}^{\infty} \frac{\sin(\pi(f_n t - k))}{\pi(f_n t - k)} x(k), \qquad (6)
\end{aligned}
$$

would exactly reconstruct a continuous signal $x_c(t)$ with only frequency content in relevant area without any deformation in this area. In practice none (or at best only part) of the future is known; different techniques exist to approximate $h(t)$. Every filtering technique then typically not only reduces high frequency signal contents, but also introduces phase lags. There are always shortcomings which the designer must be aware of.

Choice of polynomial interpolation technique In this research, several filtering techniques like Finite Impulse Response (FIR) filters, Infinite Impulse Response (IIR) filters and Cubic Polynomial Reconstruction (CPR), have been considered. Unlike FIR and IIR filters which are designed in the frequency domain, polynomial interpolation techniques can be designed in time domain. This has some advantages:

- time domain criteria can be used,
- non-linear design is possible,
- interpolation functions are continuous time functions, so they are independent of the interpolation factor.

The last point is considered important because in case of inter-connecting systems with large (> 50) ratios between their sampling rate (as in the case of host-motion computer integration), CPR interpolation techniques can still be implemented as low order filters. This is unlike the FIR-filters. Further no redesign has to done as the interpolation factor changes, unlike the IIR-filter design. In polynomial interpolation on the interval t_k to t_{k+1} of order p, a function

$$f(\xi) = \sum_{i=0}^{p} a_i \xi^i \qquad (7)$$

is defined on the local time interval ξ from 0 (t_k) to 1 (t_{k+1}). The coefficients a_i of the polynomial are chosen such that the value of the signal at the sample time points, and possibly the time derivatives thereof, satisfy certain constraints. With a third-order CPR, which was used to filter the acceleration signal, the four coefficients can for example be specified by defining the value $f(0) = x(k-1)$ and $f(1) = x(k)$ and their first time derivative $\dot{f}(1) = x(k) - x(k-1)$ and $\dot{f}(0) = x(k-1) - x(k-2)$. If a prediction of the signal $x(k+1)$ is known. the interpolation can be shifted.

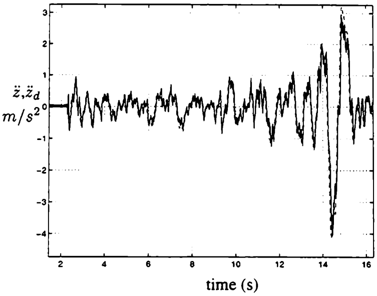


Figure 6: Example of a simulator critical maneuvre: heave acceleration response, $\ddot{z}$ and desired response, $\ddot{z}_d$, (dotted) to heavy turbulence and air pocket

With the position signal interpolation of a fifth-order (C)PR can be used to allow the accelerations (second order time derivatives) to also correspond to the desired values. In case signals from the host would arrive late, or not at all, precaution should be taken not to extrapolate the polynomials. This could run the signal off-line very quickly. Then, by taking the next sample equal to the previous one for the position the system graduaIy comes to a stop.

With CPR, the host and motion computer communicate the reference signals with a simple low-order method which has relatively little high-frequency contents. The method can easily be combined with the use of the time-derivative of the acceleration for feed forward. This 'jerk'-signal can be extracted explicitly at every time point. Predicted time points can be incorporated directly to reduce the phase lag.

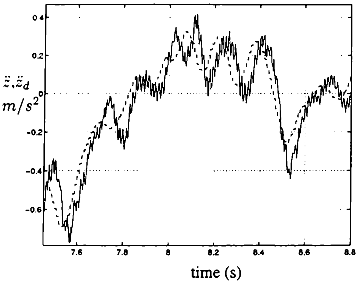


Figure 7: Example of a simulator critical maneuvre: heave acceleration response to heavy turbulence, $\ddot{z}$, and desired response, $\ddot{z}_d$, (dotted)

Experimental Results

At the KLM flight crew training centre, 31 simulator training-critical maneuvers were flown in a Boeing 747-400 simulator and the aircraft model responses registered. These maneuvers included a wide range of dynamic conditions, including take-offs, normal and hard landings, engine seizures, and response to heavy turbulence. On the basis of these recordings, the predictive model-based control was evaluated by simulation, and close similarity could be observed. A bandwidth of nearly 28 Hz was achieved (expressed as both -45-deg phase, as well as -3 dB gain). Furthermore, the simulation was observed to be robust against high-frequency parasitic system resonances (such as transmission line dynamics of the hydraulic actuators). It should be noted that increasing the sample frequency of the simulation model by up to 120 Hz would further enhance its bandwidth.

Since the simulation results were quite promising, it was decided to implement this simulation into the SRS motion platform. Five of the thirty-one maneuvers were selected, since these represented the most motion-demanding conditions.

These were:

1. *Response to maximum clear air turbulence*
   The motion system is made to move in a stochastic manner experiencing vibrations up to 10 Hz. Also an airpocket is included which requires almost full speed of actuators (.9 m/s with v_{max}=1 m/s).
2. *Taxiing*
   Fast vibrations introduced through the landing gear have to be experienced in simulator during which it has to move gradually through surge, sway and yaw motion.
3. *Landing with cross wind*
   In the air, the simulator has to experience the stochastic nature of cross wind. At landing three instances of landing gear touch down bumps require fast acceleration peak generation after which the landing slide introduces even faster vibrations up to 15 Hz.
4. *Rejected take off*
   Immediate negative surge together with tilt coordination (in which gravity is made to introduce long term breaking force) move the simulator through a wide area of operating conditions w.r.t the platform pose.
5. *Rotation during take off roll*
   Limited amount of motion also has to be introduced smoothly in the asymmetric (roll) motion. Faster resonances (before take off) during the first seconds have to be replaced by the more smooth vibrations in flying.

To test the system with the load conditions of a full operational SRS the control strategy proposed was evaluated with a dummy platform of 4000 kg on top of the SRS motion system depicted in Fig. 5. The host computer was replaced by an additional processor on the motion computer running at a lower sampling rate of 100 Hz. In the 'host'-processor the tracks for platform pose and acceleration are made available and send to the motion computer control processors at demand. The original tracks were aircraft model output. By a plain wash out these trajectories were made fit for the SRS. Actuator stroke had to be within 1.25 m and velocity within 1 m/s. To test the motion system with the control strategy these signals were considered the feasible trajectories which had to be tracked as close as possible.

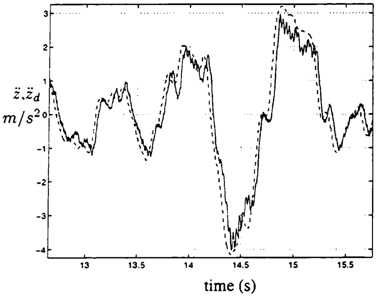


Figure 8: Example of a simulator critical maneuvre: heave acceleration response to air pocket, $\ddot{z}$, and desired response, $\ddot{z}_d$, (dotted)

Each maneuver has a duration of 16 seconds, however, due to the limited hydraulic power supply (temporarily) available at the Central Workshop of Mechanical Engineering this duration is just manageable. In Fig. 6, the measured heave response together with the reference acceleration, are depicted.

The measured heave signals correspond very closely to the desired acceleration. If it is considered on a short time frame like in Fig. 7, an almost constant lag of approximately 30 ms can be observed. The main contribution of the lag in response is due to the smoothing filter at 30 Hz (63 % rise time of 20 ms). This motivates the use of a prediction horizon of 20-30 ms, in which case the lag in response could very wel be reduced to a situation of 'virtual zero time delay'. This paper presented a number of approaches to achieve this.

Further, the figure shows that the motion system reacts without gain decrease on the stochastic nature of turbulence at least up to vibrations of ca. 8 Hz (which can be seen during the time span 8.05 s. . .8.45 s). The 'noise' in the measured signal is mainly caused by the application of servo valve dither. The top-top values remain however well below 0.01 g, which is not noticable.

If the system is driven to move upwards with a maximum velocity (e.g. in the time span 14 s. . .15 s depicted in Fig. 8), the acceleration response remains at 90 % of the required value. The basic characteristics of the actual heave acceleration are however still very well preserved. As all the actuators were required to move with almost maximum velocity the oil consumption of the system is also at maximum in this part of the maneuvre. The limited power supply at this instant in time very likely caused the slight difference during the response to a simulated air pocket.

The other maneuvres, not depicted here, show comparable characteristics. This proves that this motion system control and design strategy resulted in a high-quality system. The use of a predictive reference model based control enables a synergistic connection between host and motion computer, which will considerably reduce latencies.

Conclusions

In this paper we presented a number of approaches to reduce time delays or lags in the response of simulation motion systems. A synergistic procedure was proposed in which system knowledge of both the vehicle model to be simulated, and the motion system to be accelerated, is used to have virtually immediate response to the desired motion. By a division of sub-tasks, the modular simulation structure can be maintained.

The procedure has been validated with the SIMONA motion system and showed remarkable improvements.

Further research with a full operational simulator will be needed in order to point out exactly what delays are allowed while having *no* influence on the percieved realism of simulation. Having a simulator motion system as the SRS which is able to operate nominally with low latencies, will help enabling this research.

References

1. S.K. Advani. *The kinematic design of flight simulator motion-bases*. PhD thesis, Delft University of Technology, 1998.
2. S.K. Advani, M. van Tooren, and S. de Winter. The design of a high-performance all-composite flight simulator motion platform. In *Proceedings AIAA Flight Simulation Technologies Conference*, 1995.
3. D.E. Crochiere and L.R. Rabiner. *Multirate Digital Signal Processing*. Prentice Hall, 1983.

4. L. de Nijs. Evaluation of predictors for implementation in a feed-forward controller of a motion platform. Master's thesis, Delft University of Technology, 1997.
5. C.A. Desoer and Y.T. Wang. Linear time-invariant robust servomechanism problem: a self contained exposition. *Control and dynamic systems*, pages 81–129, 1980.
6. J. Heintze and A.J.J. van der Weiden. Inner-loop design and analysis for hydraulic actuators, with application to impedance control. *Control Engineering Practice*, pages 1323–1330, 1995.
7. S.H. Koekebakker, P.C. Teerhuis, and A.J.J. van der Weiden. Modelling and identification of a flight simulator motion system. In *Proc. 17th IASTED Conf. Modelling, Identification and Control*, pages 171–174, 1998.
8. S.H. Koekebakker, P.C. Teerhuis, and A.J.J. van der Weiden. Multiple level control of a hydraulically driven flight simulator motion system. In *Proc. IEEE Systems, Man and Cybernetics IMACS Conf. CESA'98*, pages 886–891, 1998.
9. E.A. Martin. Force and motion cueing systems. *Flight Simulation Update, American institute of Aeronautics and Astronautics*, 1995.
10. P. Piatkewitz. Reference-model-based feedforward control *for the simona motion platform*. Master's thesis, Delft University of Technology, 1997.
11. Z. Qu and D.M. Dawson. *Robust tracking contröl of robot manipulators*. IEEE-Press, 1996.
12. G. Van Schothorst. *Modelling of long-stroke hydraulic servo-systems for flight simulator motion control and system design*. PhD thesis, Delft University of Technology, 1997.
13. N. Sepehri, G.A.M. Dumont, P.D. Lawrence, and F. Sassani. Cascade control of hydraulically actuated manipulators. *Robotica*, pages 207–216, 1990.
14. J.-J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, 1991.
15. M. Tomizuka. Feedforward digital tracking controllers for motion control applications. *Advanced Robotics*, pages 575–586, 1993.
16. S. Wang and B. Chen. Model matching: optimal approximation approach. *International Journal of Control*, pages 1899–1907, 1987.

AIAA-98-4364

ON THE DESIGN OF STATICALLY BALANCED MOTION BASES FOR FLIGHT SIMULATORS

Clément M. Gosselin and Jiegao Wang
Département de Génie Mécanique
Université Laval
Québec, Québec, Canada, G1K 7P4
Tel: (418) 656-3474, Fax: (418) 656-7415
email: gosselin@gmc.ulaval.ca, jwang@gmc.ulaval.ca

Abstract

The design of statically balanced motion bases for flight simulators is addressed in this paper. First, the concept of static balancing is recalled and illustrated using a simple 1-DOF system. Then, the static balancing of motion bases is studied using two approaches, namely, using counterweights and using springs. Balancing equations are obtained for general classes of 3-, 4- and 6-DOF motion bases. In statically balanced systems, the actuators do not contribute to supporting the weight of the moving bodies. Hence the system can be in static equilibrium in any of its configurations, with zero actuator forces. Actuator forces are therefore drastically reduced, which may lead to a significant reduction in the size and power of the actuators and improve the energy efficiency.

1 Introduction

In high-performance flight simulation systems, motion bases are used to impart accelerations to the cockpit and generate motion cues. However, since the mass of the cockpit is large (often in the order of 10 metric tons), large actuator forces are usually required to support this payload, even in static mode. Indeed, actuator forces are mainly due to the weight of the system. Hence, powerful actuators are required and a significant portion of the actuation energy is spent simply to support the weight of the motion platform.

In this paper, statically balanced mechanisms are proposed for flight simulation applications. Statically balanced mechanisms are defined as those in which the actuators do not contribute to supporting the weight of the moving links, for *any* configuration. In other words, the mechanism can be brought to static equilibrium in any of its configurations, with zero actuator forces. Hence, the actuators are used only to impart accelerations to the moving links, which leads to a reduction of the size and power of the actuators and results in the improvement of the accuracy of the control.

The balancing of mechanisms has been an important research topic for several decades (see for instance [1] for a literature review). A balanced mechanism leads to better dynamic characteristics and less vibrations caused by motion. Static and dynamic balancing of planar linkages has been studied extensively in the literature (see for instance [2, 3, 4, 5, 6]).

In the context of motion simulation mechanisms and robots, static balancing is defined as the set of conditions under which the weight of the links of the mechanism does not produce any torque (or force) at the actuators under static conditions, for any configuration of the mechanism or manipulator. This condition is also referred to as *gravity compensation*. Gravity-compensated serial manipulators have been designed in [7, 8, 9, 10, 11] using counterweights, springs and sometimes cams and/or pulleys. A hybrid direct-drive gravity-compensated manipulator has also been developed in [12]. Moreover, a general approach for the static balancing of planar linkages using springs has been presented in [13]. The balancing of spatial mechanisms has also been studied, for instance in [14] and [11].

In [15], the authors have proposed statically balanced six-degree-of-freedom (DOF) parallel manipulators. Since such mechanical architectures can be also used in flight simulators, the application of the concept of static balancing to motion bases is addressed here. Two approaches of static balancing are presented, namely, i) static balancing using counterweights and ii) using springs. When the mechanism is balanced using counterweights, a mechanism with a fixed global center of mass is obtained. In other words, the static balancing is achieved in any direction of the Cartesian space of the mechanism. This próperty is useful for applications in which the mechanism is needed to be statically balanced in all directions as for instance, when a system can be installed in different orientations with respect to the gravity field. However, for motion bases, static balancing with counterweights is difficult to realize. Indeed, since the mass of the platform is very large, the counterweights required would generally be too large to be practical. Springs can be used in such instances. When springs are used, the total potential energy of the manipulator — gravitational and elastic — is set to be constant and the weight of the whole manipulator can be balanced with a much smaller total mass than when using counterweights, as pointed out in [13].

The concept of static balancing is applied here to 3-, 4- and 6-DOF motion bases and examples of balanced systems are given.

2 Static balancing of mechanisms and manipulators

Let a general spatial n-degree-of-freedom mechanism be composed of n_b moving bodies and one fixed link. Moreover, let the position vector of the center of mass of each moving body with respect to a fixed reference frame be noted $\mathbf{c}_i$ and let the mass of the ith moving body be noted m_i. The position vector of the center of mass of the mechanism with respect to the fixed frame, noted $\mathbf{c}$, can be written as

$$\mathbf{c} = \frac{1}{M}\sum_{i=1}^{n_b} m_i \mathbf{c}_i \qquad (1)$$

where M is the total mass of the moving links, i.e.,

$$M = \sum_{i=1}^{n_b} m_i \qquad (2)$$

In general, vector $\mathbf{c}$ will be a function of the configuration of the mechanism, i.e.,

$$\mathbf{c} = \mathbf{c}(\boldsymbol{\theta}) \qquad (3)$$

where $\boldsymbol{\theta}$ is the vector comprising all the joint coordinates of the mechanism.

Following this notation and if no elastic elements are used, the condition for static balancing can be written as

$$\mathbf{e}_z^T \mathbf{c} = C_t \qquad (4)$$

where C_t is an arbitrary constant and $\mathbf{e}_z$ is a unit vector oriented in the direction of gravity. In other words, the center of mass of the mechanism does not move in the direction of gravity, for any arbitrary motion of the mechanism.

When elastic elements are used, the total potential energy in the mechanism, noted V, is defined as the sum of the gravitational and elastic potential energy and can be written as

$$V = g\mathbf{e}_z^T \sum_{i=1}^{n_b} m_i \mathbf{c}_i + \frac{1}{2}\sum_{j=1}^{n_s} k_j (s_j - s_j^o)^2 \qquad (5)$$

where g is the magnitude of the gravitational acceleration, n_s is the number of linear elastic elements in the system, k_j is the stiffness of the jth elastic element, s_j is the length of the jth elastic element and s_j^o is its undeformed length. As mentioned above, when elastic elements are used, the condition for static balancing is that the total potential energy is constant, which can be written as

$$V = V_c \qquad (6)$$

where V_c is an arbitrary constant.

In articulated mechanical systems, elastic components — e.g. springs — may provide an *exact* compensation of the gravitational forces for all configurations [7, 13]. This compensation usually requires that the *effective* undeformed length of the springs be equal to zero. As shown in [13], this assumption does not present any particular problem and is generally simple to achieve in a practical system. The exact compensation can be illustrated using the simple one-degree-of-freedom example shown in Fig. 1 and taken from [13].

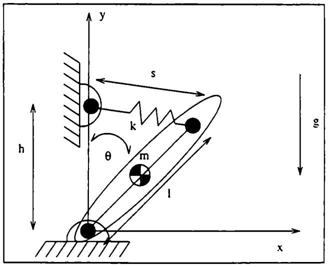


Figure 1: Simple one-degree-of-freedom statically balanced system.

In this system, a single link of mass m is rotating in a vertical plane, as illustrated in the figure. The center of mass of the link is located at a distance c from the pivot and a spring is attached to the fixed link along the vertical axis, at a distance h from the pivot, as well as to the rotating link, at a distance l from the pivot. According to eq.(5), the total potential energy in the system can be written as

$$V = mgc\cos\theta + \frac{1}{2}k(s - s^o)^2 \qquad (7)$$

where k is the stiffness of the spring while s and s^o are its length and its undeformed length, respectively. From the law of cosines, one can write

$$s^2 = h^2 + l^2 - 2hl\cos\theta \qquad (8)$$

Moreover, if the undeformed length of the spring is equal to zero, i.e.,

$$s^o = 0 \qquad (9)$$

then, eq.(8) can be substituted directly in eq.(7), which leads to

$$V = (mgc - khl)\cos\theta + \frac{1}{2}k(h^2 + l^2) \qquad (10)$$

Hence, the total potential energy of the system will be constant if and only if one has

$$mgc = khl \qquad (11)$$

or, in other words, if the stiffness of the spring is chosen such that

$$k = \frac{mgc}{hl} \qquad (12)$$

The resulting statically balanced system will therefore be in static equilibrium for *any* configuration, as long as the orientation of the base of the mechanism with respect to the gravity field is not changed. The latter observation is important: when springs are used, static balancing will be obtained for *one given orientation of the base* with respect to the gravity field. On the other hand, when static balancing is achieved using only a redistribution of the masses and the use of counterweights, it usually leads to mechanisms which are statically balanced for *any* orientation and magnitude of the gravity. For instance, in the above example, a balanced mechanism would be obtained by removing the spring and adding a counterweight such that the center of mass of the link would be moved to the center of the pivot. Such a mechanism would be statically balanced for any orientation of the gravity field.

In the following sections, the balancing equations for motion systems with 3-, 4- and 6-DOF will be obtained using the general approach presented above.

3 Balancing of 3-DOF Motion Bases

3.1 Balancing Equations for Mechanisms without Springs

A spatial 3-DOF motion base is illustrated schematically in Figs. 2 and 3. It consists of three identical legs connecting the base to the platform. Each of these legs consists of a revolute joint attached to the base, a first moving link, a second revolute joint, a second moving link and a spherical joint attached to the platform. Moreover, since the revolute joints of a given leg are parallel, each leg is constrained to move in a plane.

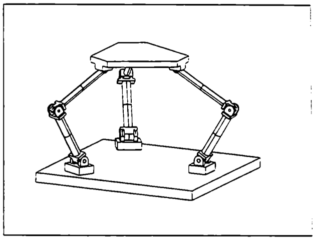


Figure 2: CAD model of a 3-DOF motion base.

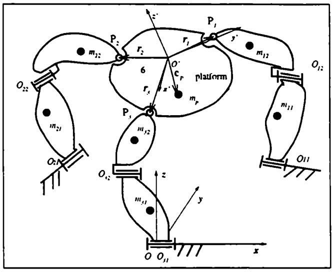


Figure 3: Schematic representation of the 3-DOF motion base.

The position of the center of mass of the mechanism can be written as

$$M\mathbf{r} = m_p\mathbf{r}_p + \sum_{i=1}^{3}(m_{i1}\mathbf{r}_{i1} + m_{i2}\mathbf{r}_{i2}) \qquad (13)$$

where M is the total mass of all moving links of this system, $\mathbf{r}$ is the position vector of its global center of mass, $\mathbf{r}_p, \mathbf{r}_{i1}$, and $\mathbf{r}_{i2}$ are the position vectors of the moving platform, the first and the second moving link of the ith leg and m_p, m_{i1} and m_{i2} are the corresponding masses. The position vectors of the center of mass of each moving link can be written as

$$\begin{aligned}\mathbf{r}_p &= \mathbf{Q}_{31}\mathbf{l}_{31} + \mathbf{Q}_{32}\mathbf{l}_{32} + \mathbf{Q}(\mathbf{c}_p - \mathbf{r}_3) && (14)\\ \mathbf{r}_{i1} &= \mathbf{r}_{io} + \mathbf{Q}_{i1}\mathbf{c}_{i1}, \quad i = 1,2,3 && (15)\\ \mathbf{r}_{i2} &= \mathbf{r}_{io} + \mathbf{Q}_{i1}\mathbf{l}_{i1} + \mathbf{Q}_{i2}\mathbf{c}_{i2}, \quad i = 1,2,3 && (16)\end{aligned}$$

where $\mathbf{r}_{io}$ $(i = 1,2,3)$ are constant vectors connecting the origin of the fixed frame to a point defined by the intersection of the axis of the first revolute joint and the plane of motion of the ith leg. Vectors $\mathbf{l}_{i1}$ and $\mathbf{l}_{i2}$ are the position vectors connecting the origins of consecutive body frames while vectors $\mathbf{c}_{i1}$ and $\mathbf{c}_{i2}$ are the position vectors of the center of mass of the first and second moving link of the ith leg, respectively. Moreover, $\mathbf{c}_p$ and $\mathbf{r}_i (i = 1,2,3)$ are respectively the position vectors from the origin of the frame attached to the platform to the center of mass of the platform and to the points of attachment of the legs to the platform. All the vectors mentioned above — except $\mathbf{r}_{io}$ — are expressed in a moving local reference frame and hence rotation matrices associated with the revolute joints are defined as $\mathbf{Q}_{i1}$ and $\mathbf{Q}_{i2}$.

Substituting eqs.(14)–(16) into eq.(13) and eliminating dependent variables using the closed-loop kinematic constraints, one finally obtains:

$$M\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \qquad (17)$$

where

$$\begin{aligned}r_x = {} & D_{w1}q_{11} + D_{w2}q_{12} + D_{w3}q_{13}\\ & + D_{w4}\sin\gamma_3\sin\theta_{31}\\ & + D_{w5}\sin\gamma_3\sin\theta_{32} + D_{w6}\sin\gamma_1\sin\theta_{12}\\ & + D_{w7}\sin\gamma_2\sin\theta_{22} - \sum_{i=1}^{3} D_{wi8}\sin\gamma_i\cos\theta_{i1}\\ & + \sum_{i=1}^{3} D_{wi9}\sin\gamma_i\cos\theta_{i2} + D_{0x} \qquad (18)\\ r_y = {} & D_{w1}q_{21} + D_{w2}q_{22} + D_{w3}q_{23}\\ & - D_{w4}\cos\gamma_3\sin\theta_{31}\\ & - D_{w5}\cos\gamma_3\sin\theta_{32} - D_{w6}\cos\gamma_1\sin\theta_{12}\\ & - D_{w7}\cos\gamma_2\sin\theta_{22} + \sum_{i=1}^{3} D_{wi8}\cos\gamma_i\cos\theta_{i1}\\ & + \sum_{i=1}^{3} D_{wi9}\cos\gamma_i\cos\theta_{i2} + D_{0y} \qquad (19)\\ r_z = {} & D_{w1}q_{31} + D_{w2}q_{32} + D_{w3}q_{33} + D_{w4}\cos\theta_{31}\\ & + D_{w5}\cos\theta_{32} + D_{w6}\cos\theta_{12}\\ & + D_{w7}\cos\theta_{22} + \sum_{i=1}^{3} D_{wi8}\sin\theta_{i1}\\ & + \sum_{i=1}^{3} D_{wi9}\sin\theta_{i2} + D_{0z} \qquad (20)\end{aligned}$$

in which

$$\begin{aligned}D_{w1} &= m_p(x_p - a_3) + \sum_{i=1}^{2}\frac{a_i - a_3}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1})\\ D_{w2} &= m_p(y_p - a_3) + \sum_{i=1}^{2}\frac{b_i - b_3}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1})\\ D_{w3} &= m_p(z_p - a_3) + \sum_{i=1}^{2}\frac{c_i - c_3}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1})\\ D_{w4} &= l_{31}[m_p + \sum_{i=1}^{3}\frac{m_{i1}z_{i1} + m_{i2}l_{i1}}{l_{i1}}]\\ D_{w5} &= l_{32}m_p + m_{32}z_{32} + l_{32}\sum_{i=1}^{2}\frac{m_{i1}z_{i1} + m_{i2}l_{i1}}{l_{i1}}\\ D_{w6} &= m_{12}z_{12} - \frac{l_{12}}{l_{11}}(m_{11}z_{11} + m_{12}l_{11})\\ D_{w7} &= m_{22}z_{22} - \frac{l_{22}}{l_{21}}(m_{21}z_{21} + m_{22}l_{21})\\ D_{wi8} &= m_{i1}y_{i1}, \quad i = 1,2,3\\ D_{wi9} &= m_{i2}y_{i2}, \quad i = 1,2,3\\ D_{0x} &= \sum_{i=1}^{2}[\frac{x_{io}}{l_{i1}}(m_{i1}x_{i1}\cos\gamma_i - m_{i2}l_{i1})]\\ D_{0y} &= \sum_{i=1}^{2}[\frac{y_{io}}{l_{i1}}(m_{i1}y_{i1}\sin\gamma_i - m_{i2}l_{i1})]\\ D_{0z} &= -\sum_{i=1}^{2}[\frac{z_{io}}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1})]\end{aligned}$$

where angles γ_i are constant architecture parameters, q_{ij} is the ijth element of the rotation matrix associated with the Cartesian orientation of the platform and θ_{ij} is the coordinate associated with the jth revolute joint of the ith leg.

Therefore, in order for the global center of mass of the mechanism to be fixed, the coefficients of the kinematic (joint or Cartesian) variables, namely, D_{wi} $(i = 1,\ldots,7), D_{wi8}$ $(i = 1,\ldots,3)$ and D_{wi9} $(i = 1,\ldots,3)$ must vanish. Hence, one obtains the conditions for static balancing as follows

$$\begin{aligned}D_{wi} &= 0, \quad i = 1,\ldots,7 && (21)\\ D_{wi8} &= 0, \quad i = 1,2,3 && (22)\\ D_{wi9} &= 0, \quad i = 1,2,3 && (23)\end{aligned}$$

Clearly, if these conditions are satisfied, the center of mass of the mechanism will be fixed in all directions and therefore the mechanism will be balanced for any direction of the gravity vector.

3.2 Balancing Equations for Mechanisms with Springs

When springs are used, the potential energy of the mechanism includes the potential energy associated with gravity and the elastic potential energy stored in the springs. Hence, the expression of the total potential energy can be written as

$$V = Mgr_z + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{q} k_{ij}e_{ij}^2 \qquad (24)$$

where k_{ij} and e_{ij} are respectively the stiffness and the deformation of the jth spring of the ith leg while n is the number of legs and q is the number of springs used in each leg. Similarly to what was presented in [9, 13], it is assumed here that the undeformed length of the springs is zero, in order to be able to obtain perfect balancing. As explained in the aforementioned references, this constraint can easily be implemented in practical systems by using simple mechanical elements.

In order to use springs to balance the mechanism, a special architecture (similar to what was used in [13] for planar linkages) is proposed for the legs. As represented in Fig. 4, a parallelogram four-bar linkage is used instead of the first link of the ith leg. Kinematically, this architecture is equivalent to the one presented in the previous section but the special architecture is important for the second spring of the ith leg to play a role in the static balancing of the mechanism. From the geometry of the mechanism and since the undeformed lengths of the springs are zero, one can write

$$e_{i1}^2 = h_{i1}^2 + d_{i1}^2 - 2h_{i1}d_{i1}\cos\theta_{i1}, \quad i = 1,2,3 \qquad (25)$$

$$e_{i2}^2 = h_{i2}^2 + d_{i2}^2 - 2h_{i2}d_{i2}\cos\theta_{i2}, \quad i = 1,2,3 \qquad (26)$$

where h_{ij} and d_{ij} are, on the ith leg, the distance from the jth joint to the fixed and moving attachments of the spring, respectively (Fig. 4).

Substituting the results in eq.(24) and eliminating some of the joint variables using the kinematic constraints, one finally obtains

$$\begin{aligned} V = {} & D_{s1}q_{31} + D_{s2}q_{32} + D_{s3}q_{33} + D_{s4}\cos\theta_{31} \\ & + D_{s5}\cos\theta_{32} + D_{s6}\cos\theta_{12} + D_{s7}\cos\theta_{22} \\ & + \sum_{i=1}^{3}(D_{si8}\sin\theta_{i1}) + \sum_{i=1}^{3}(D_{si9}\sin\theta_{i2}) \\ & + D_0 \end{aligned}$$

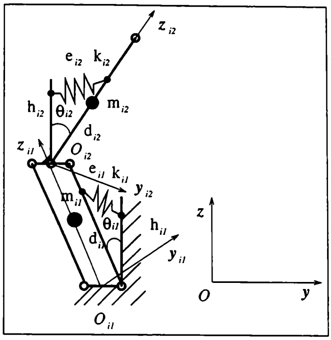


Figure 4: Geometric architecture of the ith leg.

where

$$\begin{aligned} D_{s1} &= m_pg(x_p - a_3) + g\sum_{i=1}^{2}\frac{a_i - a_3}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1}) \\ & + \frac{k_{11}h_{11}d_{11}}{l_{11}}(a_3 - a_1) + \frac{k_{21}h_{21}d_{21}}{l_{21}}(a_3 - a_2) \\ D_{s2} &= m_pg(y_p - b_3) + g\sum_{i=1}^{2}\frac{b_i - b_3}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1}) \\ & + \frac{k_{11}h_{11}d_{11}}{l_{11}}(b_3 - b_1) + \frac{k_{21}h_{21}d_{21}}{l_{21}}(b_3 - b_2) \\ D_{s3} &= m_pg(z_p - c_3) + g\sum_{i=1}^{2}\frac{c_i - c_3}{l_{i1}}(m_{i1}z_{i1} + m_{i2}l_{i1}) \\ & + \frac{k_{11}h_{11}d_{11}}{l_{11}}(c_3 - c_1) + \frac{k_{21}h_{21}d_{21}}{l_{21}}(c_3 - c_2) \\ D_{s4} &= l_{31}[m_pg + g\sum_{i=1}^{3}\frac{m_{i1}z_{i1} + m_{i2}l_{i1}}{l_{i1}} \\ & - (\frac{k_{11}h_{11}d_{11}}{l_{11}} + \frac{k_{21}h_{21}d_{21}}{l_{21}} + \frac{k_{31}h_{31}d_{31}}{l_{31}})] \\ D_{s5} &= [l_{32}m_pg + m_{32}gz_{32} + l_{32}g\sum_{i=1}^{2}\frac{m_{i1}z_{i1} + m_{i2}l_{i1}}{l_{i1}}] \\ & - [l_{32}(\frac{k_{11}h_{11}d_{11}}{l_{11}} + \frac{k_{21}h_{21}d_{21}}{l_{21}}) + k_{32}h_{32}d_{32}] \\ D_{s6} &= [m_{12}gz_{12} - \frac{l_{12}}{l_{11}}(m_{11}gz_{11} + m_{12}gl_{11})] \\ & + l_{12}(\frac{k_{11}h_{11}d_{11}}{l_{11}} - \frac{k_{12}h_{12}d_{12}}{l_{12}}) \\ D_{s7} &= [m_{22}gz_{22} - \frac{l_{22}}{l_{21}}(m_{21}gz_{21} + m_{22}gl_{21})] \end{aligned}$$

$$
\begin{aligned}
& +l_{22}(\frac{k_{21}h_{21}d_{21}}{l_{21}} - \frac{k_{22}h_{22}d_{22}}{l_{22}}) \\
D_{si8} &= m_{i1}gy_{i1}, \quad i = 1,2,3 \\
D_{si9} &= m_{i2}gy_{i2}, \quad i = 1,2,3 \\
D_0 &= \frac{1}{2}\sum_{i=1}^{3}[k_{i1}(h_{i1}^2 + d_{i1}^2) + k_{i2}(h_{i2}^2 + d_{i2}^2)] \\
& +k_{11}h_{11}d_{11}z_{1o} + k_{21}h_{21}d_{21}z_{2o} \\
& -\sum_{i=1}^{3}[\frac{z_{io}}{l_{i1}}(m_{i1}gz_{i1} + m_{i2}gl_{i1})]
\end{aligned}
$$

Hence, for the potential energy to be constant for any configuration, one must have

$$
\begin{aligned}
D_{si} &= 0, \quad i = 1,\ldots,7 && (27) \\
D_{si8} &= 0, \quad i = 1,2,3 && (28) \\
D_{si9} &= 0, \quad i = 1,2,3 && (29)
\end{aligned}
$$

which are the balancing conditions. It can be observed that the balancing conditions now involve the magnitude of the gravitational acceleration g, as it should. It can also be noticed that, in a context of design, the balancing conditions of the manipulator with springs offer more freedom to the designer since the stiffness of the springs can be adjusted, which increases the number of free variables in the equations.

3.3 Examples

Two examples of balanced mechanisms are represented schematically in Figs.5 and 6. As can be realized from the figures, large counterweights have to be mounted on the distal links of the legs in order to balance the mechanism if no springs are used and the total mass would be increased. However, the center of mass of the resulting mechanism remains fixed for any configuration of the mechanism, which means that the static torques at the actuators are zero for any configuration and for any orientation of the gravity vector with respect to the base of the mechanism. On the other hand, when springs are used, no counterweights are required but the mechanism will be balanced for one specific magnitude and direction of the gravity vector. In motion simulation applications, this limitation is generally not important.

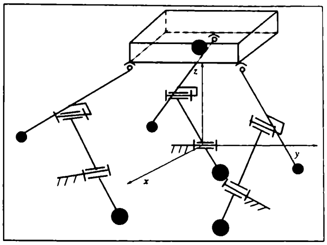


Figure 5: Statically balanced 3-DOF motion base with counterweights.

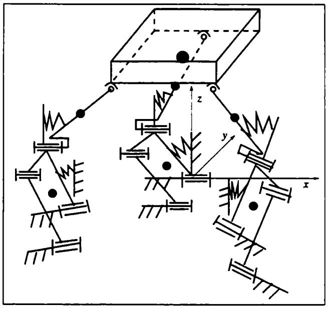


Figure 6: Statically balanced 3-DOF motion base with springs.

4 Balancing of 4-DOF Motion Bases

4.1 Balancing Equations for Mechanisms without Springs

Two types of 4-DOF motion bases with prismatic actuators are illustrated in Figs. 7 and 8. The mechanism of Fig. 7 consists of eleven link bodies. namely, a fixed base, a moving platform, four actuated legs consisting of two links and an unactuated special leg consisting of only a single link. The mechanism of Fig. 8 includes an additional body in the 5th leg. The four actuated legs are connected to the base via Hooke joints and to the platform via spherical joints. The two bodies of these legs are connected by actuated prismatic joints. The fifth leg is connected to the base via a revolute joint. This special unactuated leg is connected to the platform via a spherical joint in the first mechanism and via a Hooke joint in the second mechanism. Similar mechanisms could

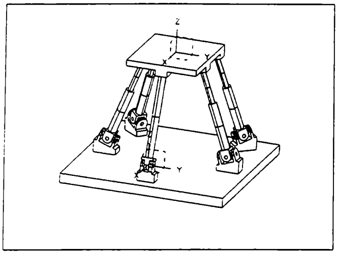


Figure 7: First type of 4-DOF motion base.

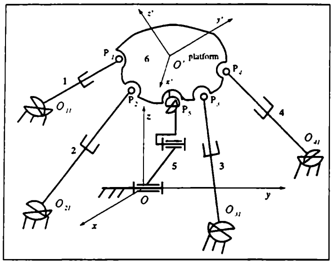


Figure 8: Schematic representation of the second type of 4-DOF motion base.

be built using revolute actuators.

The first mechanism is now considered. In order to simplify the study, it is assumed that the center of mass of each of the two links constituting the ith leg lie on line $O_{i1}P_i$, as represented in Fig. 9. This assumption is reasonable, since all other designs will only lead to a more complicated architecture of the leg without any advantage.

Points C_{iu} and C_{il} are respectively the locations of the center of mass of the upper and lower link of the ith leg. Lengths l_{iu} and l_{il} denote the distances from C_{iu} to P_i and from O_{i1} to C_{il} respectively, while ρ_i is the variable length of the ith actuated leg, namely, the distance from O_{i1} to P_i.

One can write

$$\mathbf{p}_i = \mathbf{p}_5 + \mathbf{Q}(\mathbf{p}_i^{'} - \mathbf{p}_5^{'}), \quad i = 1, \ldots, 4 \qquad (30)$$

where $\mathbf{p}_i (i = 1, \ldots, 4)$ is the position vector of point P_i expressed in the fixed coordinate frame, $\mathbf{p}_i^{'}$ is the position vector of point P_i expressed in the moving

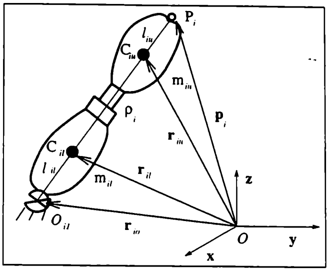


Figure 9: The two link bodies of ith leg of the mechanisms.

coordinate frame,

$$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, i = 1, \ldots, 4 \quad \mathbf{p}_i^{'} = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}, i = 1, \ldots, 5 \qquad (31)$$

and $\mathbf{p}_5$ is the position vector of point P_5 expressed in

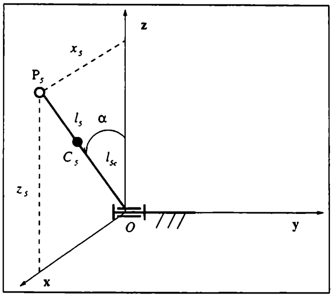


Figure 10: Geometry of the fifth leg of the first mechanism.

the fixed coordinate frame, as represented in Fig. 10, which can be expressed as

$$\mathbf{p}_5 = \begin{bmatrix} l_5 \sin\alpha \\ 0 \\ l_5 \cos\alpha \end{bmatrix} \qquad (32)$$

If the coordinates of point O_{i1} expressed in the base coordinate frame are given as $(x_{io}, y_{io}, z_{io})(i = 1, \ldots, 4)$, one can compute the position vector of the center of mass of the 5th leg, as well as the position of the center of mass of the upper and lower links of the ith leg, namely, the position vectors of points C_5, C_{iu} and C_{il}, as represented in Figs. 10 and 9 as

$$\mathbf{r}_{iu} = \mathbf{p}_i - \frac{l_{iu}}{\rho_i}(\mathbf{p}_i - \mathbf{r}_{io}), \quad i = 1, \ldots, 4 \qquad (33)$$

$$\mathbf{r}_{il} = \mathbf{p}_i - (1 - \frac{l_{il}}{\rho_i})(\mathbf{p}_i - \mathbf{r}_{io}), \quad i = 1, \ldots, 4 \qquad (34)$$

$$\mathbf{r}_5 = \mathbf{p}_5(\frac{l_{5c}}{l_5}) \qquad (35)$$

where $\mathbf{r}_5, \mathbf{r}_{iu}$ and $\mathbf{r}_{il}$ are respectively the position vectors of the center of mass of the 5th leg, the upper link of the ith leg and the lower link of the ith leg. Vector $\mathbf{r}_{io}$ is the position vector of point O_{i1}, l_5 is the length of the 5th leg and l_{5c} is the distance from O to C_5, the center of mass of the fifth leg.

The global center of mass of the mechanism can then be written as

$$M\mathbf{r} = m_p\mathbf{r}_p + m_5\mathbf{r}_5 + \sum_{i=1}^{4}(m_{il}\mathbf{r}_{il} + m_{iu}\mathbf{r}_{iu}) \qquad (36)$$

where M is the total mass of all moving links of the mechanism, $\mathbf{r}$ is the position vector of the global center of mass, m_p, m_5, m_{il} and m_{iu} are respectively the masses of the platform, the 5th leg, the upper link and lower link of the ith leg.

Substituting the appropriate expressions into eq.(36), one then obtains

$$M\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \qquad (37)$$

where

$$\begin{aligned} r_x &= [D_8 + l_5 \sum_{i=1}^{4}(\frac{D_i}{\rho_i})]\sin\alpha \\ &+[D_5 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(a_i - a_5)]q_{11} \\ &+[D_6 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(b_i - a_5)]q_{12} \\ &+[D_7 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(c_i - a_5)]q_{13} \\ &-\sum_{i=1}^{4}(z_{io}\frac{D_i}{\rho_i}) + \sum_{i=1}^{4} m_{il}z_{io} \qquad (38) \\ r_y &= [D_5 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(a_i - a_5)]q_{21} \\ &+[D_6 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(b_i - a_5)]q_{22} \\ &+[D_7 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(c_i - a_5)]q_{23} \\ &-\sum_{i=1}^{4}(z_{io}\frac{D_i}{\rho_i}) + \sum_{i=1}^{4} m_{il}z_{io} \qquad (39) \\ r_z &= [D_8 + l_5 \sum_{i=1}^{4}(\frac{D_i}{\rho_i})]\cos\alpha \\ &+[D_5 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(a_i - a_5)]q_{31} \\ &+[D_6 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(b_i - a_5)]q_{32} \\ &+[D_7 + \sum_{i=1}^{4}\frac{D_i}{\rho_i}(c_i - a_5)]q_{33} \\ &-\sum_{i=1}^{4}(z_{io}\frac{D_i}{\rho_i}) + \sum_{i=1}^{4} m_{il}z_{io} \qquad (40) \end{aligned}$$

and where

$$\begin{aligned} D_i &= l_{il}m_{il} - l_{iu}m_{iu}, \quad i = 1, \ldots, 4 \\ D_5 &= (x_p - a_5)m_p + \sum_{i=1}^{4}(a_i - a_5)m_{iu} \\ D_6 &= (y_p - b_5)m_p + \sum_{i=1}^{4}(b_i - b_5)m_{iu} \\ D_7 &= (z_p - c_5)m_p + \sum_{i=1}^{4}(c_i - c_5)m_{iu} \\ D_8 &= l_5(m_p + \sum_{i=1}^{4} m_{iu}) + l_{5c}m_5 \end{aligned}$$

A sufficient condition for the global center of mass of the mechanism to be fixed is that the coefficients noted $D_i (i = 1, \ldots, 8)$ vanish. Therefore, one obtains the conditions for the static balancing as follows

$$D_i = 0, \quad i = 1, \ldots, 8 \qquad (41)$$

Similar conditions are obtained for the mechanism of Fig. 8. The detailed expressions are not given here because of space limitation.

4.2 Balancing Equations for Mechanisms with Springs

As mentioned above, when springs are used, the potential energy of the mechanism is given by eq. (24). In the present case, a spring is attached to the fifth leg only.

Hence, the total potential energy in the system is similar to the expression given above and the balancing conditions are obtained as

$$D_i = 0, \quad i = 1,\dots,7 \qquad (42)$$

$$D_8 - 2k_5h_5d_5 = 0 \qquad (43)$$

where the coefficients are as defined for the mechanism without springs. Similar conditions are obtained for the mechanism of Fig. 8. The detailed expressions are not given here because of space limitation.

4.3 Examples

Two examples of balanced mechanisms are represented schematically in Figs.11 and 12. As can be realized from Fig. 11, a large counterweight is necessary to balance the mechanism. On the other hand, when a spring is used (Fig. 12), no counterweights are required but the mechanism will be balanced for one specific magnitude and direction of the gravity vector. In motion simulation applications, this limitation is generally not important.

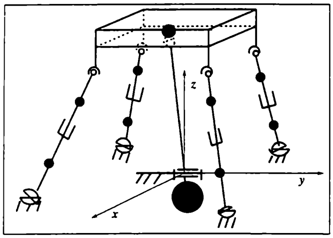


Figure 11: Balanced 4-DOF motion base using counterweights.

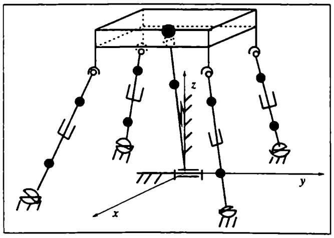


Figure 12: Balanced 4-DOF motion base using springs.

5 Balancing of 6-DOF Motion Bases

Following the approach derived above, it is possible to obtain the balancing conditions for the 6-DOF motion system represented in Fig. 13. In this system, each of the legs consists of a fixed revolute actuator, a first moving link a Hooke joint a second moving link and a spherical joint attached to the platform.

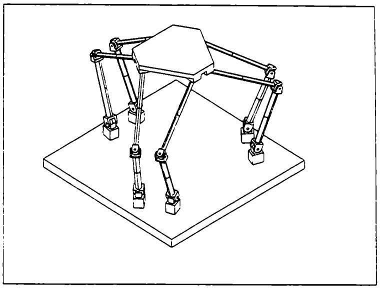


Figure 13: 6-DOF motion base with revolute actuators.

The detailed balancing conditions are given in [15]. An example of balanced mechanism without springs is given in Fig. 14. As can be realized from the numerical results and from the figure, large counterweights are necessary to balance the mechanism.

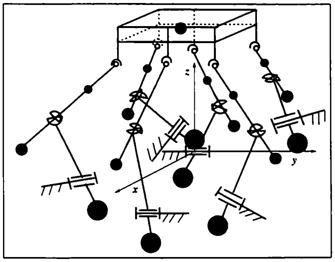


Figure 14: Balanced 6-DOF motion base using counterweights.

If springs are used, different leg architectures are possible. Two possible choices are illustrated schematically in Figs. 15 and 16. The balancing conditions are given in [15], for each of these cases. An example of balanced system using the second leg architecture is illustrated in Fig. 17.

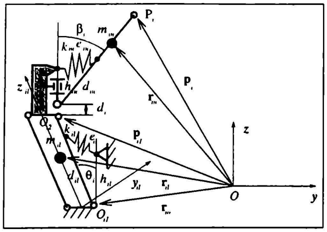


Figure 15: First possible kinematic architecture of the ith leg.

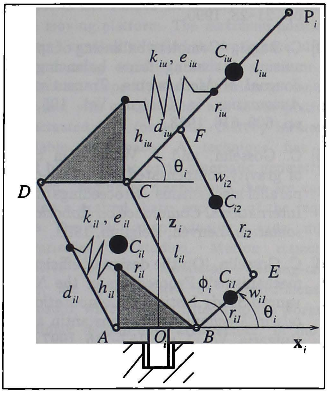


Figure 16: Second possible architecture of the ith leg.

6 Example of a Spherical 3-DOF Architecture

Finally, an example of a motion simulation device based on a 3-dof 366 spherical parallel mechanism is represented in Fig. 18. This motion simulator is being developed for a single-user facility. In this particular device, the center of mass of the platform is very close to the center of rotation of the system and therefore, no springs are used for the balancing. The balancing equations for this system can be readily obtained and are given in [16].

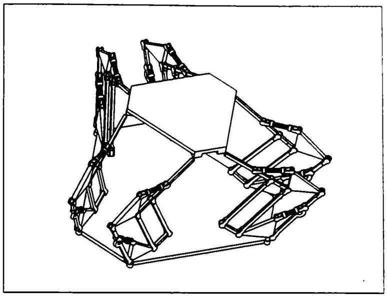


Figure 17: Balanced 6-DOF motion base using springs.

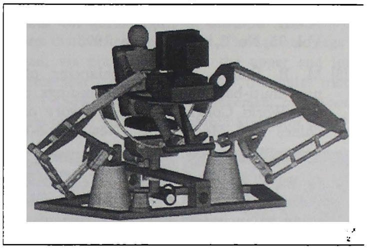


Figure 18: A motion simulation device based on the 3-DOF 366 spherical parallel mechanism.

7 Conclusion

The design of statically balanced motion bases for flight simulators has been addressed in this paper. First, the concept of static balancing has been recalled and illustrated using a simple 1-DOF system. Then, the static balancing of motion bases has been studied and balancing equations have been derived for classes of 3-, 4- and 6-DOF motion bases. In statically balanced systems, the actuators do not contribute to supporting the weight of the moving bodies. Hence the system can be in static equilibrium in any of its configurations, with zero actuator forces. Actuator forces are therefore drastically reduced, which may lead to a significant reduction in the size and power of the actuators and improve the energy efficiency. A 6-DOF small-scale balanced system is currently being built to demonstrate the concept. This system will have a payload of a few kilograms. The concept will then by applied to a full-scale system with a payload of the order of one metric ton.

Acknowledgements: This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would also like to thank Gabriel Côté and Nicolas Pouliot for their help with the preparation of some of the figures.

References

[1] G.G. Lowen, F.R. Tepper and R.S. Berkof, 'Balancing of linkages – an update', *Mechanism and Machine Theory*, Vol. 18, No. 3, pp. 213–220, 1983.

[2] E. N. Stevenson Jr., 'Balancing of machines', *ASME Journal of Engineering for Industry*, Vol. 95, No. 2, pp. 650–656, 1973.

[3] M. R. Smith, 'Optimal balancing of planar multi-bar linkages', *Proceedings of the 5th World Congress on the Theory of Machines and Mechanisms*, New-Castle-Upon-Tyne, pp. 142–149, 1975.

[4] C. Bagci, 'Shaking force balancing of planar linkages with force transmission irregularities using balancing idler loops', *Mechanism and Machine Theory*, Vol. 14, No. 4, pp. 267–284, 1979.

[5] F. Gao, 'Complete shaking force and shaking moment balancing of 17 types of eight-bar linkages only with revolute pairs', *Mechanism and Machine Theory*, Vol. 26, No. 2, pp. 179–206, 1991.

[6] Z. Ye and M.R. Smith, 'Complete balancing of planar linkages by an equivalence method', *Mechanism and Machine Theory*, Vol. 29, No. 5, pp. 701–712, 1994.

[7] R.H. Nathan, 'A constant force generation mechanism', *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, No. 4, pp. 508–512, 1985.

[8] J.M. Hervé, 'Device for counter-balancing the forces due to gravity in a robot arm', United States Patent 4,620,829, May 1986.

[9] D.A. Streit and B.J. Gilmore, 'Perfect spring equilibrators for rotatable bodies', *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 111, No. 4, pp. 451–458, 1989.

[10] N. Ulrich and V. Kumar, 'Passive mechanical gravity compensation for robot manipulators', Proceedings of the *IEEE International Conference on Robotics and Automation*, Sacramento, pp. 1536–1541, 1991.

[11] G.J. Walsh, D.A. Streit and B.J. Gilmore, 'Spatial spring equilibrator theory', *Mechanism and Machine Theory*, Vol. 26, No. 2, pp. 155–170, 1991.

[12] H. Kazerooni and S. Kim, 'A new architecture for direct drive robots', Proceedings of the *IEEE Int. Conference on Robotics and Automation*, Philadelphia, pp. 442–445, 1988.

[13] D.A. Streit and E. Shin, 'Equilibrators for planar linkages', Proceedings of the *ASME Mechanisms Conference*, Chicago, Vol. DE-25, pp. 21–28, 1990.

[14] C. Bagci,, 'Complete balancing of space mechanisms – shaking force balancing', *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 105, No. 12, pp. 609–616, 1983.

[15] C. Gosselin, and J. Wang, 'On the design of gravity-compensated six-degree-of-freedom parallel mechanisms', Proceedings of the *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[16] C. Gosselin, 'On the design of efficient parallel mechanisms', *Proceedings of the NATO Advanced Study Institute: Computational Methods in Mechanics*, St. Konstantin and Elena, Bulgaria, Vol. 1, pp. 157–186, 1997.

AIAA-98-4367

A Parametric Study of Optimized Six-Degrees-of-Freedom Motion Systems

S.K. Advani[1], J.F. Albronda[2]

International Centre for Research in Simulation, Motion and Navigation Technologies SIMONA,
Delft University of Technology, Anthony Fokkerweg 1, 2629 HS, Delft, The Netherlands, s.advani@lr.tudelft.nl

Abstract

The kinematic geometry of a six-degrees-of-freedom motion system determines its ability to position and orient the moving platform. The maximum positioning and orienting capability, called the workspace, dictates the motion cueing capability of the simulator. The designer is free to specify the mechanism geometry by the shape and size of the upper and lower platforms, and the actuated legs (or jacks), thereby influencing the available workspace. A technique has been developed by which the basic kinematic geometry can be optimized, leading to mechanisms that are capable of achieving a desired shape and size of the required workspace. The effect of introducing a wide range of design variables is studied. Motion trajectories representative of 31 simulator training-critical maneuvers, are approximated by a six-dimensional hyper-ellipsoid, which is then fitted into the workspace of a candidate motion-base architecture. In the first optimization, the size of this hyper-ellipsoid is maximized, while in a second case, its shape is held constant and the motion-base size is minimized. In every case, a specified minimum dexterity of the mechanism is maintained, to keep the actuator loads within reasonable limits. Re-arranging the motion-base geometry can lead to substantial improvements in the positioning and orienting capability of the motion base. In particular, allowing changes to the fixed (base) frame of the mechanism can significantly increase the ability to tailor the workspace to a desired form.

Introduction

Flight simulator motion-base geometries are most commonly configured as Stewart Platforms[14]. This arrangement is comprised of a base-frame, six prismatic actuator legs (or jacks), and an upper moving platform that carries the payload. The actuated legs are attached, via gimbal joints, to the upper and lower platforms near the vertices of their triangular frames. Figure 1 shows the general arrangement of a Stewart Platform on the SIMONA Research Simulator (SRS).

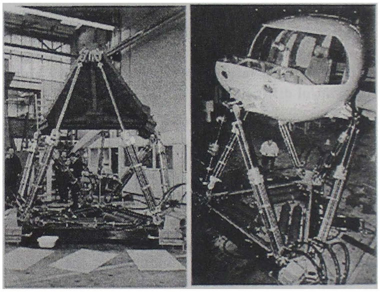


Figure 1 - Stewart Platform motion-base architecture (SIMONA Research Simulator), with test platform (left), and all-composite flight-deck (right)

In a Stewart Platform arrangement, the locations of the six upper and six lower gimbal joints can be mapped on circles. The gimbal pairs are separated by a fixed offset. See Figure 2. The motion of the legs, all six of which are identical, is constrained by their minimum and maximum lengths. These constraints dictate the mechanism's cueing capability by defining the kinematic envelope that the upper platform can achieve with respect to the inertial reference frame. Larger workspace in a given direction generally allows a

[1] Director of SIMONA, Assistant Professor, Member AIAA

[2] Research Assistant

longer cue duration. Increasing proportionally the size of *all* of the members of the Stewart Platform simultaneously extends the maximum translational capabilities of the upper platform, yet will not affect its rotational limits.

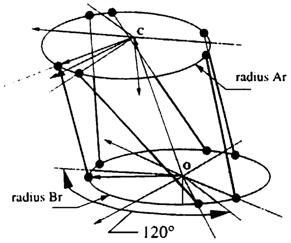


Figure 2 - Kinematic representation of the typical circular layout of a conventional Stewart Platform mechanism

Deviations from the standard circular (upper and lower platform) arrangements are possible, however the platform must never achieve a pose that is at, or close to *singularities*. In this situation, the ratio of actuator displacements to the resulting platform displacements is very low, meaning that the positioning accuracy may suffer, the mechanical loads can become very high, or the control difficult.

Goal of this research

Once a moving-base simulator is available, a motion drive algorithm is applied, and heuristically tuned to generate motion cues that attempt to represent those that the aircraft pilot would experience[9]. This motion drive algorithm reads the outputs of the simulated aircraft model, and generates jack length commands for the motion cueing mechanism. The motion platform may never exceed the limits of the actuators, but should simultaneously provide reasonable motion cues to the pilot. If the kinematic limits are reached however, the pilot may detect a bump in the motion, known as a false cue[6, 7]. If the lower-frequency motion cues are attenuated, there is a risk that they will not be detected by the pilot. Applying adaptive motion-cueing algorithms, in order to solve both of the above, can however introduce non-linearities, which may then be perceived as characteristics of the simulated vehicle[11].

Thus, one a motion-base has been built, we can only hope that the motion cueing can be tuned to what the pilot should perceive[12]. If, however, the *spatial* requirements of the motion platform in its *motion-cueing* role, were better known *prior* to its construction, then the architecture of that mechanism could be tailored to provide the *required* motion cueing workspace.

Previous research[1, 2, 10] showed that this tailoring is possible through optimization of the mechanism architecture, by specifying the relative shape of the workspace, and adjusting the mechanism to accommodate this shape. This approach made use generic shapes, as well as approximated aircraft responses generated from a flight simulation model of a Boeing 747-400, fed through a unity-gain classical washout filter to predict the simulator trajectories[4].

In this paper, a wide range of design parameters will be studied. The design variables (or parameters specified by the designer that can vary during the optimization) will be classified, along with geometric constants, into groups called Architectures. The effects of *two* washout filters on the resulting mechanisms will be shown.

Simulator workspace requirements

The workspace *required* for a particular simulation task, represented by the integral sum of these motion trajectories, depends on the vehicle properties, as well as the maneuvers that must be simulated. In this research, aircraft mathematical model responses were registered during a range of dynamic flight conditions. The measurements were carried out with the help of a Level D qualified full-flight simulator with a mathematical model representing the Boeing 747-400, and flown by a qualified pilot. Thirty-one training-critical maneuvers were recorded. The time histories of the aircraft specific forces and angular rates were then passed through two forms of a linear classical motion-drive algorithm in which the input signal limiting was removed[4]. (The washout algorithm had previously been tuned to a similar aircraft type[13]). The two forms were, respectively, a unity-gain algorithm (Washout 1, or WF-1), a half-gain algorithm (Washout 2, or WF-2).

Figure 3 shows the translational and angular time histories mapped onto two-dimensional degree-of-freedom projections. The ellipses circumscribe the maximum excursions in each pair of degree-of-freedom, and serve as the weighting factors for the design synthesis phase. Due to the six required degrees-of-freedom, fifteen two-dimensional cross-section mappings are required. Finally, all of these are combined to create a "hyper-ellipsoid" in six-space.

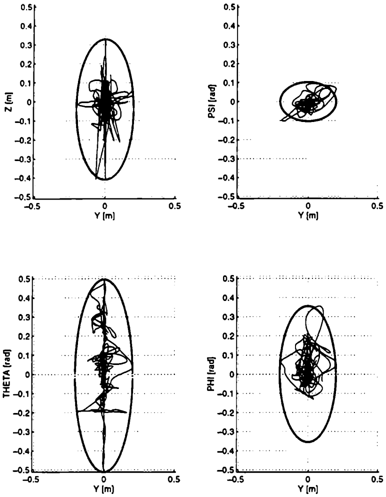


Figure 3 - Platform centroid trajectories are mapped onto all cross-sectional planes. Note that only 4 of the 15 mappings are shown here.

Workspace Weighting Factors

The aircraft/washout-specific weighting factors are given in Table 1, and were obtained by inscribing the trajectories resulting from the filtered aircraft motions, to approximate the required workspace.

Table 1 - Objective function weighting factors

	ρ_x	ρ_Y	ρ_z	ρ_ψ	ρ_θ	ρ_ϕ
WF-1	0.821	0.204	0.370	0.103	0.504	0.355
WF-2	0.342	0.072	0.188	0.051	0.251	0.177

Design Strategy

In its fully general form, the kinematic layout of a six degrees-of-freedom synergistic platform requires the specification of the six upper and six lower gimbal attachment point in 3-D space, as well as the minimum and the maximum actuator lengths for each of the six actuators. This leaves the designer with 6*3 + 6*3 + 6*2 = 48 free design variables from which to select. Varying any one of these variables will influence the final cueing performance of the design.

Free Design Variables

All the mechanisms investigated were symmetrical about the X-Z plane, however a number of design degrees-of-freedom, or free design variables, were introduced, with the goal of achieving a family of motion-bases having a simulation task-tailored workspace. These free variables are now described.

A. Elliptical Platform/Base-Frame

The location of the upper and lower attachment points on their respective platforms can be generalized from a circle (in the conventional Stewart platform) to an ellipse, as shown in Figure 4.

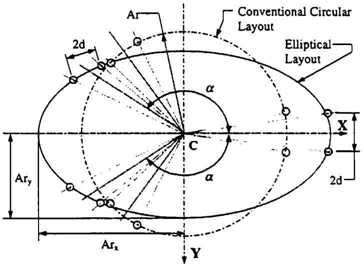


Figure 4 - Layout of standard (circular) and elliptical (shaded) platform (or base-frame). Gimbals are mapped along the boundary of the ellipse (rather than circle), and angle α can range between 90° and 170°. Distance 2d is held constant.

The size and aspect ratio of these ellipses were made variable, thus resulting in two design variables per platform. The separation between each pair of attachment points was fixed at the minimum physically achievable distance (2d for the upper platform and 2p for the lower platform) A sensitivity study showed that this yielded the maximum workspace. One pair of attachment points was fixed to lie symmetrically on the x-axis, while the other 2 pairs were located symmetrically at an angle α from the x-axis (and correspondingly an angle β on the lower platform). The angles α and β constitute 2 additional design variables. A planar elliptical platform (or base-frame) constitutes therefore three design variables: Ar_x. Ar_y and α.

B. Non-Planar Base Frame

The base frame gimbals, as suggested earlier, can be positioned vertically above or below a reference origin, as shown in Figure 5.

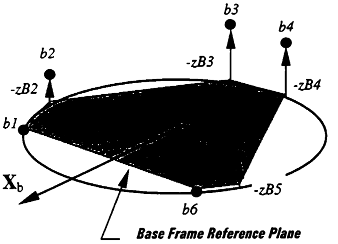


Figure 5 - Layout of non-planar base-frame

By rules of symmetry,

$$\begin{aligned} zB_1 &= zB_6 \\ zB_2 &= zB_5 \\ B_3 &= zB_4 \end{aligned} \qquad [1]$$

C. General Base-Frame

The aforementioned non-planar base-frame can be allowed even greater freedom during the optimization, by removing the requirement to map the gimbals onto an ellipse. Instead, the locations of the gimbals are made fully general, as shown in Figure 6.

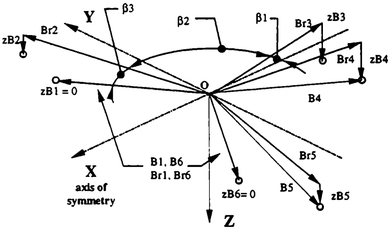


Figure 6 - General Base-Frame, representing four design variables.

D. Variable Actuator Lengths

The actuator lengths Q can also used as 'design variables. For a given general simulator actuator design, the maximum and minimum lengths, and hence the stroke, can be related through

$$Q_{max} = 2 \cdot Q_{min} - 0.981 \qquad [2]$$

This represents a family of actuators with identical mechanical hardware, except that the cylinders and pistons are cut to differing lengths. Equation [2] is determined empirically and is based on existing motion base hardware. Through Eq. [2], if all actuators are identical, the introduction of the actuator length introduces one free design variable. If non-identical actuators are considered, one free design variable per actuator *pair* is required, thus

$$\begin{aligned} Q_{min,1} &= Q_{min,6} \\ Q_{min,2} &= Q_{min,5} \\ Q_{min,3} &= Q_{min,4} \\ Q_{min,1} &\neq Q_{min,2} = Q_{min,3} \end{aligned} \qquad [3]$$

However, to prevent the actuators from achieving excessive stroke lengths, the following constraint was applied.

$$Q_{min,1} + Q_{min,2} + Q_{min,3} \leq 3 \cdot Q_{min,Avge} \qquad [4]$$

where $Q_{min,Avge}$ =2.131m.

Motion-Base Architectures

The optimization studied a systematic increase in the number of design variables, allowing increasing deviation from the conventional Stewart platform, and leading to a more general class of flight simulator motion base. Each of these groupings of design variables is termed an architecture.

Table 2 summarizes all the design variables and indicates their allowed ranges.

Table 3 describes the architectures studied in this research.

Table 2 - Summary of selected design variables

Design Variable	Description of Variable	Min. Limit	Max. Limit
Ar_x	moving platform elliptical semi-axis length along X_p	1.50 m	3.00 m
Ar_y	moving platform elliptical semi-axis length along Y_p	1.50 m	3.00 m
Br_x	base-frame elliptical semi-axis length along X_b	1.50 m	3.00 m
Br_y	base-frame elliptical semi-axis length along Y_b	1.50 m	3.00 m
α	platform gimbal angle	90 °	170 °
β	base frame gimbal angle	90 °	170 °
Qmin $_{1,6}$	min. length, act.1 & 6	1.381 m	2.981 m
Qmin $_{2,5}$	min length, act. 2 & 5	1.381 m	2.981 m
Qmin $_{3,4}$	min length, act.3 & 4	1.381 m	2.981 m
zB $_{1,6}$	vertical position of base-frame gimbals 1 & 6	0 (reference position)	
zB $_{2,5}$	vertical position of base-frame gimbals 2 & 5	-0.50 m	0.50 m
zB $_{3,4}$	vertical position of base-frame gimbals 3 & 4	-0.50 m	0.50 m

Table 3 - Description of optimized Architectures and number of design variables

Arch	Constraints Description and Free Design Variables	Nv
0	Stewart Platform (Not optimized further) Ar=1.60m, Br=1.65m, P=0.10m, D=0.30m, Qmin=2.081m	0
1	Upper and lower platforms remain circular Ar, Br free during optimization	2
2	Circular platforms, with free gimbal orientations and six equal actuator lengths. Ar, Br, α, β, and Qmin free.	5
3	Elliptical platforms with equal actuators. Ar, Br, α, β, Qmin free.	7
4	Elliptical platforms with unequal actuators. Ar, Br, α, β, Qmin $_{1,6}$, Qmin $_{2,5}$, Qmin $_{3,4}$ free.	9
5	Elliptical platforms with equal actuators, but vertical offset in base-frame. Ar, Br, α, β, Qmin zB_{25}, zB_{34} free.	9
6	Circular platform, general base-frame. Ar, Br_1, Br_2, Br_3, α, $β_{16}$, $β_{25}$, $β_{34}$, zB_{25}, zB_{34} Qmin, free.	10

Optimization of the Motion-Base

An optimization technique utilizing a *Linearly-Constrained Quadratic Programming* sub-problem was developed in the previous work. Like most optimization methods reported in literature[8], this iterative technique minimizes an objective function while remaining within specified constraints. At each step, the new solution is checked against the optimality conditions for the true problem to determine whether another step must be taken. The optimization is schematically shown in Figure 7.

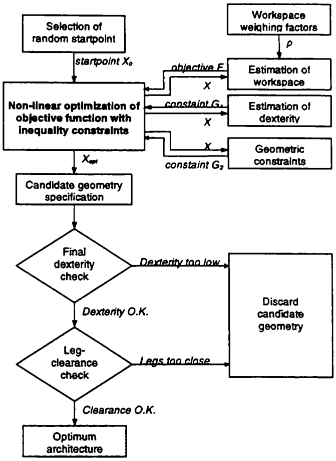


Figure 7 - Simulator motion-base architecture optimization procedure[10]

Specification of the Objective Function

A hyper-ellipsoid, created by merging the two-dimensional mappings, like those shown in Figure 3, into a six-dimensional mathematical function, was used to describe the required motion volume. Its mathematical form is defined as,

$$\left(\frac{X-X_0}{\rho_X}\right)^2+\left(\frac{Y-Y_0}{\rho_Y}\right)^2+\left(\frac{Z-Z_0}{\rho_Z}\right)^2+\left(\frac{\psi-\psi_0}{\rho_\psi}\right)^2+\left(\frac{\theta-\theta_0}{\rho_\theta}\right)^2+\left(\frac{\varphi-\varphi_0}{\rho_\varphi}\right)^2\leq 1 \qquad [5]$$

where X_0,..., φ_0 denote the neutral position of the motion base (i.e. when all actuators are at their mid-stroke). This also corresponds to the midpoint of the hyper-ellipsoid. The *weighting factors* ρ_X,..., ρ_φ denote the ellipsoid semi-axis length in each direction. If the motion requirements are available as a set of trajectory maps (as in Figure 3), the hyper-ellipsoid can be fitted to the data that contain the midpoints and weighting factors. The objective function to be maximized is then:

$$\overline{X}^2+\overline{Y}^2+\overline{Z}^2+\overline{\psi}^2+\overline{\theta}^2+\overline{\varphi}^2=R \quad [6]$$

where $\overline{X}$, $\overline{Y}$,... are generalized coordinates, denoting a *scaled*, non-dimensional distance from the platform's neutral position, for example,

$$\overline{X}=\frac{X-X_0}{\rho_X} \quad [7]$$

The factor R, called the ***weighted radius of the hyper-ellipsoid***, scales the ellipsoid proportionally without changing its shape. The objective function given by eq.(4) attempts to find the ***maximum weighted radius*** R_{max}, describing the largest hyper-ellipsoid which just fits into the workspace. $R_{max} \geq 1$implies that the workspace of the motion-base will indeed contain the design trajectories. Because the motion-base must be symmetric about the X-Z plane, it follows that,

$$Y_0=\psi_0=\varphi_0=0 \quad [8]$$

Figure 8 shows this concept for three cross-sections of the *translational* workspace, namely X-Y, X-Z and Y-Z. Here, the weighting factors are $\rho_X=\rho_Y=1$ and $\rho_Z=0.5$, and the critical ellipse is located in the X-Z plane.

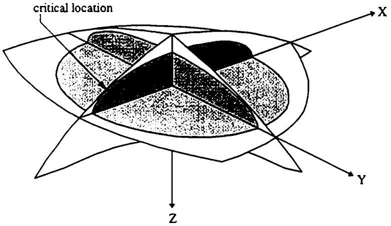


Figure 8 - Three inscribed ellipses in cross-sectional planes of the workspace.

In order to define the inscribed ellipse for each cross-section, the boundary of the workspace in that cross-section must first be determined. A numerical approach was developed to find the boundary, by moving the platform from its neutral position in a certain direction, thereby changing the leg lengths. At some point, one of these leg lengths will reach its specified length limit, thus restraining the platform motion. Then, the workspace boundary can be estimated by repeatedly searching for the boundary, each time in a different direction (the *offset* direction), while always starting from the same initial position. Figure 9 shows how a boundary estimate for the X-Z plane is found with 16 discrete offset directions.

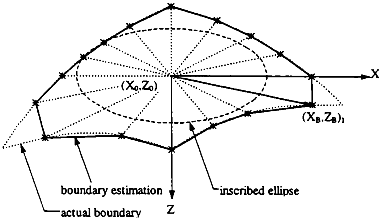


Figure 9 - Estimate for X-Z motion boundary based on 16 discrete boundary searches.

Problems can arise with the controllability of the platform when the platform pose varies substantially for very small motions of the actuators. The mathematical limit of this situation is called a *singularity* of the platform, and must be avoided throughout the workspace, as it can lead to malfunctions and failure of the platform. Even nearness to singularities should be avoided, as they can result in excessively high actuator loads.

A mathematical term, commonly used in the field of robotics to express the conditioning (poor condition implies proximity to a singularity) is referred to as *dexterity*. The dexterity varies throughout the workspace, and it is the ***minimum*** dexterity that we are interested in constraining. Design experience suggested that a value of 0.2 must always be maintained.

During each cycle of the optimization, the minimum dexterity is checked in 64 platform poses corresponding to the minimum/maximum extensions of the six actuators. Once the optimum has been found, a more thorough check of the minimum dexterity is performed, by dividing each of the actuator lengths into ten sections, and evaluating the dexterity at each of the 10^6 combinations. Then, the minimum absolute distance between all the legs is also computed. If this minimum ever falls below a specified level (in this case 15 cm), the program indicates that there is a potential *leg-crossing* situation and the candidate geometry is rejected.

Maximum Workspace Optimization

Figure 10 shows the influence that the specified architectures (specified sets of design variables) have on the achievable objective function. The corresponding value of the objective function achieved indicates the largest hyper-ellipsoid, with relative proportions described by the workspace weighting factors, that can be fitted into the workspace of a particular mechanism. The base-line case, Architecture 0, in Figure 10, represents a conventional Stewart Platform. The optimizations that follow show then the potential improvement when increasing design freedom is allowed.

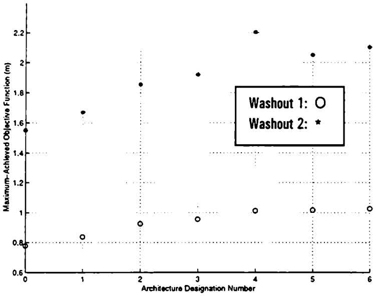


Figure 10 - Optimized Objective Function R as a function of the selected Architecture

In each washout case, an objective function of at least 1 is required in order to ensure that the mechanism can achieve the simulation trajectories without limiting any of its actuators.

From Figure 10, the achieved (fitted) objective function indeed depends on the washout that is used to generate the weighting factors. Note that in the case of Washout 2, which has a filter input gain of 0.5, an excessively large workspace results, given the design variables and their geometric constraints. In every case, its objective function exceeds 1. Conversely, for Washout 2, a relatively small mechanism would be required to accomplish the simulation task. Washout 1, on the other hand, demands a much larger workspace, and only Architectures 4, 5 and 6 all provide enough workspace to accomplish the task. Thus, given the design variables and the workspace criteria, each of these three architectures permits a suitable workspace for the required simulation task. Moreover, the designer may then select from any of these last three architectures, depending on other (economic) factors. On this point, it is interesting to see that Architecture 4, in which the actuator pairs have unequal lengths, the objective function corresponding to Washout 2 is clearly the largest. On the other hand, less complicated (and perhaps cheaper) modifications, namely to the base-frame (Architecture 6), also provide a satisfactory improvement in the workspace.

The differences between the objective functions obtained for Washoutl and Washout 2 show also the highly non-linear nature of this problem and, hence, the need for a parametric study prior to concluding which architiecture is best matched for a particular R.

Minimum Leg-Length Optimization

The optimization was modified such that the workspace (R_{min}) became a constraint (along with minimum dexterity of 0.2, as previously), and the ***minimum leg-length*** (Q_{min}) became the objective function. Whereas, from Figure 10, it is clear that the reduction in the leg-length would not be major in the case of Washout 1, a greater potential exists for the tailoring of the workspace when Washout 2 is used.

Table 4 - Optimization results (minimum actuator leg-length optimization) for Architectures 4, 5 and 6. Washout 1 (unity gains) applied.

Architecture 4	Architecture 5	Architecture 6
Achieved Objective Function (Actuator Length Q_{min})		
2.071m	2.106 m	2.098 m
Corresponding Actuator Stroke		
1.090 m (average)	1.125 m	1.117 m
Optimum Design Variables		
Arx = 1.500 m	Arx = 1.500 m	α = 113°
Ary = 1.500 m	Ary = 1.500 m	β_1 = 10°
Brx = 1.500 m	Brx = 1.500 m	β_2 = 102°
Bry = 2.514 m	Bry = 2.884 m	β_3 = 4.8°
α = 121°	α = 119°	R_1 = 1.5 m
β = 121°	β = 114°	R_2 = 2.472 m
Q_{min1} = 2.157 m	Q_{min} = 2.106 m	R_3 = 2.121 m
Q_{min2} = 2.002 m	Z_{25} = 0.500 m	Q_{min} = 2.098 m
Q_{min3} = 2.055 m	Z_{34} = 0.228m	Z_{25} = 0.49 m
		Z_{34} = 0.355 m
Platform Neutral Pose		
X_n = 0.355 m	X_n = 0.573 m	X_n = -0.383m
Z_n = -2.008 m	Z_n = -1.797 m	Z_n = -1.919m
θ_n = -9.8°	θ_n = -3°	θ_n = 15.1°

Table 4 gives the final values of the design variables and at the global optimum for Architectures 4, 5 and 6 following the minimum leg-length optimization, and when Washout 1 (Classical Washout with unity gains) is applied. The same is shown for the reduced-gain (Washout 2) in Table 5.

Table 5 - Optimization results (minimum actuator leg-length optimization) for Architectures 4, 5 and 6. Washout 2 (reduced gains) applied.

Architecture 4	Architecture 5	Architecture 6
Achieved Objective Function (Q_{min})		
1.498 m	1.488 m	1.487 m
Corresponding Actuator Stroke		
0.517m (average)	0.507 m	0.506 m
Optimized Design Variables		
Arx = 1.500 m	Arx = 1.500 m	α = 94°
Ary = 1.500 m	Ary = 1.500m	β_1 = 13°
Brx = 1.500 m	Brx = 1.500 m	β_2 = 81.5°
Bry = 1.523 m	Bry = 1.797 m	β_3 = 2.5°
α = 122°	α = 106.6°	R_1 = 1.602m
β = 97°	β = 115.5°	R_2 = 1.977m
Q_{min1} = 1.500 m	Q_{min} = 1.488 m	R_3 = 1.519m
Q_{min2} = 1.522 m	Z_{25} = 0.164m	Q_{min} = 1.487m
Q_{min3} = 1.472 m	Z_{34} = -0.487 m	Z_{25} = 0.207m
		Z_{34} = -0.128m
Platform Neutral Pose		
X_n = -0.490 m	X_n = -0.089 m	X_n = -1.14 m
Z_n = -1.209m	Z_n = -1.238 m	Z_n = -1.08 m
θ_n = -11.1 °	θ_n = -14.45 °	θ_n = -3.0 °

Discussion of Results

The relevant points of the above tables are now discussed:

- In Architectures 4 and 5, the upper platform tends toward a circular form (Arx = Ary), however in all cases, the geometric constraint of 1.5 is active. Meanwhile, the lower platform tends to be elliptical, with a wider base (Ary > Arx), and Arx again striking its lower limit of 1.5. Therefore, the upper platform constraint applied to Architecture 6 (which forces it to have a circular shape) could be considered a reasonable simplification.

- The allowance of vertical offsets in the lower gimbal attachment points (Architectures 5 and 6), as opposed to unequal actuators (Architecture 4), achieves similar results and may be an economically-preferable solution.

- While the minimum leg-length optimization significantly shrinks the actuator stroke, its full potential is likely limited by the geometric constraints of, for example, the upper platform. Reducing its size further would require careful consideration of the moving load geometry, however, for smaller simulator applications, the optimizations could be continued even further.

- In the optimized architectures, the neutral pose is not specified, but is an outcome of the optimization. Non-zero values of θ_n would imply that the simulator cab would have to be designed to provide a *floor* with zero tilt in the neutral pose. Since one is also free to chose which direction is fore and which is aft, this tilting of the platform gimbal plane may offer advantages, for example, with respect to the simulator visual display field-of-view, or structural design.

Graphical Representations:

The optimized mechanisms of Table 4 (for Washout 1), when the minimum leg-length optimization is applied, are graphically presented in Figure 11. Also shown for comparison, in the first row, is the SRS Stewart Platform. Note that the base frames (shaded) in each optimized case tend to be spread in the Y rather than in the X-direction, to permit the larger X-displacement required (see the weighting factors, Table 1).

Workspace Plots

Figure 12 shows a subset of the 15 planar cross-sections through the six-dimensional workspace of three platforms, as well as the corresponding maximum inscribed ellipses. Again, the minimum leg-length optimization has been applied here. The platforms shown are the base-line (SRS Stewart Platform), and then Architecture 6 for Washouts 1 and 2, respectively. The reduced workspace for Washout 2 is clearly evident in the third column.

In all optimized cases, the workspace tends to "shrink" around the required ellipse. (This also raises concerns, due to the elliptical approximation, about the actual actuator extensions during simulation maneuvers. This is addressed below). In none of the cases (including those not shown) did the workspace fit the ellipse exactly. This is because (a) the geometry and characteristics of the parallel mechanism do not necessarily lend themselves to this shape, and (b) the dexterity constraint, always being active, creates workspace that may never be used during operation but, if encountered, the mechanism will remain stable.

Actuator Displacements

While the elliptical approximation of the required workspace (which is actually a six-dimensional volume) is used to define the minimum workspace requirements, the platform centroid trajectories that are originally used to size the ellipses may "protrude" outside the actual workspace. Practically, this means that the mechanism may yet encounter an actuator limit during some maneuver. Therefore, the actuator lengths during these maneuvers should be checked finally, as shown in Figure 13, against their maximum and minimum allowable operational limits (dotted lines). The result for two of the six actuators is shown for all 31 recorded flight maneuvers, passed through Washout 1. In the first row, for the SRS Stewart Platform, the washout gain would have to be reduced (to about 75 percent) to prevent actuator limiting.

In the second row, the same washout is shown with Architecture 6. Limiting appears to be completely avoided for these two actuators. In actual fact, in this entire study, the problem of limiting was almost always, but not completely, avoided (see row 3). Examination of the anticipated motions is therefore vital, and it is therefore beneficial to perform the optimization for more than one Architecture. While the elliptical approximation of the trajectories is a good engineering simplification to a complex problem, care must be taken to ensure that the design will not eventually be restricted. In actual fact, however, the maneuvers will never be identical to those used in the weighting process, and some limited software limiting may be needed. Optimizing the initial pose of the mechanism has shown further improvement to the fitting of the trajectories[5].

Conclusions

Through this new motion-base design approach, a closer match between the required workspace and that provided by the mechanism can be created. Inclusion of the minimum dexterity ensures that the mechanism will always avoid proximity to singularities that can make platform control difficult. The parametric study of optimized designs has shown that a number of architectures, and not just one combination of design variables, yield suitable solutions. This allows the designer to consider other practical issues. Should all of the actuators require the same stroke, then Architectures 5 and 6 will suffice, and require then unconventional base-frames with vertical offsets. Architecture 4 requires three pairs of actuators having (slightly) different lengths. Moreover, the washout directly influences the outcome. In fact, changes to the washout gains can, through this design technique, influence the economic advantages of a particular mechanism. For example, despite the stroke limitations of electrically-driven actuators, a mechanism using these can be configured to still provide a pre-determined level of cueing. In any case, the length of the actuators, hydraulic or electric, should be minimized to maintain high natural frequencies.

References

1. Advani, S.K., "The Kinematic Design of Flight Simulator Motion-Bases". Ph.D. Thesis, Delft University of Technology, April 1998. ISBN 90-407-1672-2.
2. Advani, S.K., Nahon, M., Haeck, N., Albronda, J.F., "Optimization of Six-Degrees-of-Freedom Flight Simulator Motion Systems". AIAA Modeling and Simulation Technologies Conference, August 1997, New Orleans. AIAA-97-3504-CP.
3. Advani, S.K. et al, "SIMONA - A Reconfigurable and Versatile Research Facility". AIAA Modeling and Simulation Technologies Conference, New Orleans, August 1997, AIAA-97-3809-CP.
4. Albronda, J.F., "In-Situ Determination of Aircraft Responses During Execution of Flight-Simulator Critical Maneuvers". Prelim. Thesis Report, TU Delft, Fac. of Aerospace Eng., 1997.
5. Albronda, J.F., "Optimization of the Initial Pose of Six-DOF Flight Simulator Motion-Bases". Masters thesis report, Faculty of Aerospace Eng., Delft Univ. of Technology, 1998.
6. Baarspul, LR-248, "The Generation of Motion Cues on a Six-Degrees-of-Freedom Motion System", Faculty of Aerospace Engineering Report LR-248, 1977.
7. Baarspul, M., "A Review of Flight Simulation Techniques", From Progress in Aerospace Sciences, Vol. 27, No. 1, 1990, ISSN 0376-0421/90
8. Gill, P., Murray, W., and Wright, M.H., Practical Optimization, Academic Press, 1981.
9. Grant, P.R., Reid, L.D., "Motion Washout Filter Tuning: Rules and Requirements" In AIAA Flight Simulation Technologies Conference, AIAA-95-3408-CP, Baltimore, MD, August 1995.
10. Haeck, N., "Optimization of Six-Degrees-of-Freedom Flight Simulator Motion Systems". Masters thesis report, Faculty of Aerospace Engineering, TU Delft, 1997.
11. Hosman, R.J.A.W., "Pilots Perception and Control of Aircraft Motion". Ph.D. Thesis, Delft University of Technology, November 1996
12. Nahon, M. and Reid, L.D., "Simulator Motion-Drive Algorithms: A Designer's Perspective". Journal of Guidance, AIAA, Vol. 13, No. 2, 1989, pp. 356-362.
13. Reid, L.D., and Nahon, M., "Response of Airline Pilots to Variations in Flight Simulator Motion Algorithms". AIAA Journal of Aircraft, Vol. 25, No. 7, pp. 639-648, July 1988.
14. Stewart, D., "A platform with six-degrees-of-freedom", in Proc. Inst. of Mech. Eng., v180, part 1, no. 5, 1965-1966, pp. 371-386.

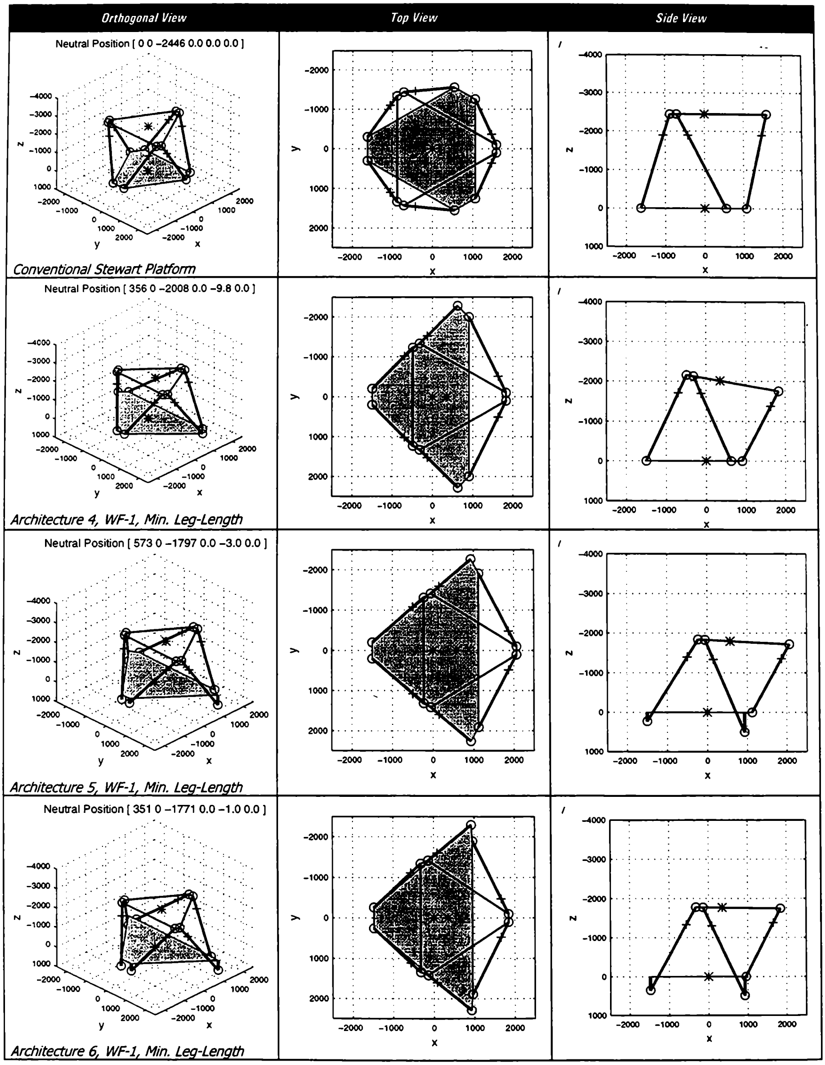


Figure 11 - Graphical representation of the candidate architectures,

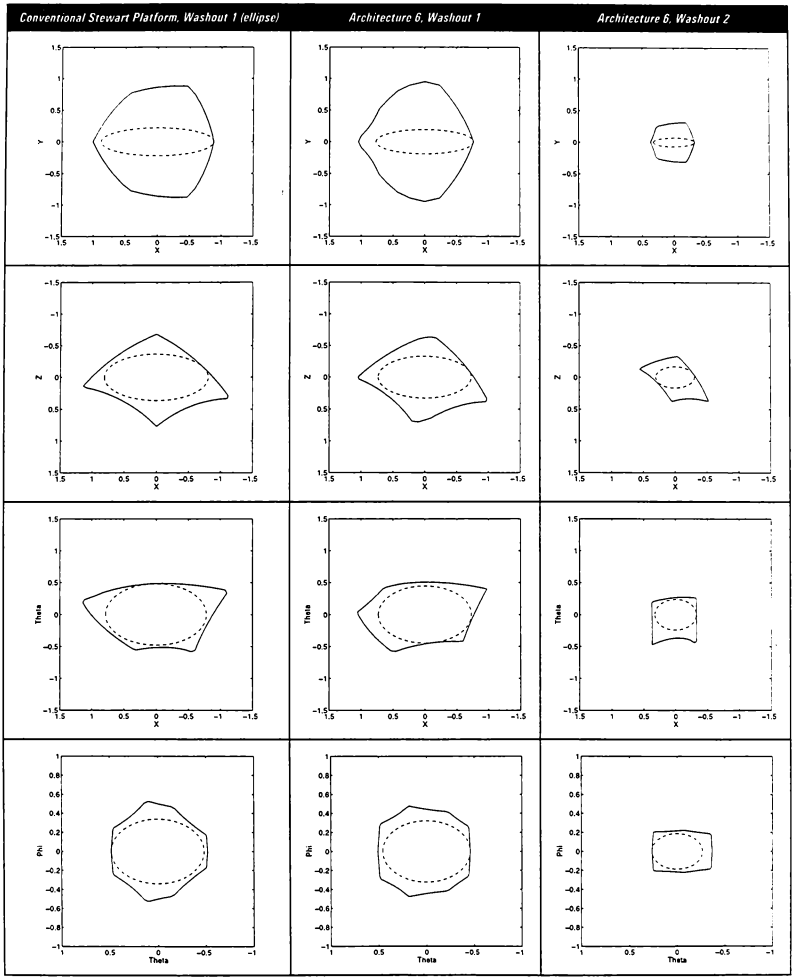
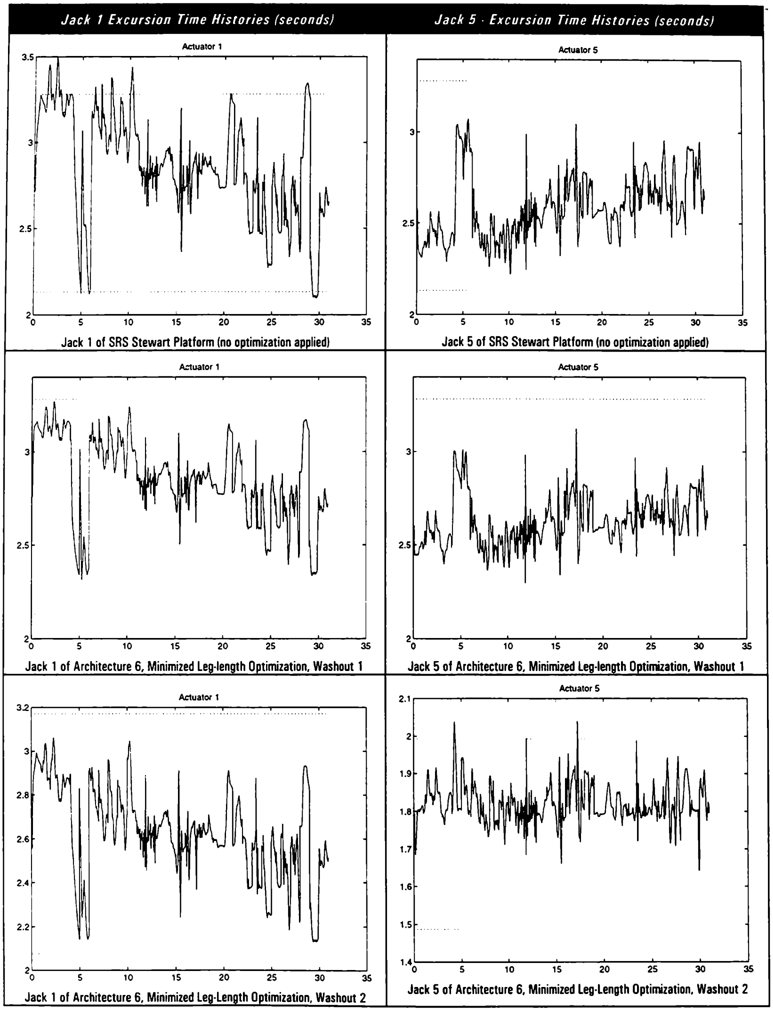


Figure 12 · Cross-sections of the workspace in X-Y direction, X-Z direction, X-θ direction and φ-θ direction and inscribed ellipses for these cross-sections


Jack 1 Excursion Time Histories (seconds)

Jack 5 · Excursion Time Histories (seconds)

Jack 1 of SRS Stewart Platform (no optimization applied)

Jack 5 of SRS Stewart Platform (no optimization applied)

Jack 1 of Architecture 6, Minimized Leg-length Optimization, Washout 1

Jack 5 of Architecture 6, Minimized Leg-length Optimization, Washout 1

Jack 1 of Architecture 6, Minimized Leg-Length Optimization, Washout 2

Jack 5 of Architecture 6, Minimized Leg-Length Optimization, Washout 2

Figure 13 · Actuator lengths during 31 training maneuvers, for SRS (top), and Architecture 6 (WF-1 and WF-2, consecutively below). Actual jack limits (upper and lower) in each case are given by the dashed lines

AIAA-98-4565

NEW METHODS FOR INCREASED FIDELITY AND SAFETY OF SIMULATION MOTION PLATFORMS

Judy A. Carmein
Senior Controls and Simulation Engineer

Allen J. Clark
Applications Specialist

MTS Systems Corporation, Advanced Engineering Solutions Division
14000 Technology Drive, Eden Prairie, Minnesota, 55344-2290, USA

Abstract

Desirable performance characteristics of motion platforms used for simulation include high bandwidth, low phase lags and minimal cross-axis coupling. Because people occupy simulators, the need for safety cannot be over-emphasized. Using standard feedback control techniques there is a trade-off between performance and stability, where performance is usually sacrificed for greater stability and safety.

Recent advances in real-time computational power coupled with in-depth knowledge of the dynamic and hydraulic characteristics of motion platforms have been used to build a new system architecture. This architecture increases the performance of multi-degree of freedom motion platforms without sacrificing safety. The new system includes a model based feedforward path that determines the required servovalve opening for each actuator to achieve the desired motion. A lightly tuned feedback path runs in parallel. Because the feedback path is so softly tuned, control stability is not a concern. This paper explains the basic system strategy and presents data obtained from a recently built six degree of freedom Stewart platform with a table and payload mass of 2500 lbm. Results show a linear phase in all six degrees of freedom associated with a pure time delay between 9 and 14 ms and less than +/- 2 db magnitude variation to 10 Hz. In addition, the new method improved the overturning moment disturbance rejection when compared to standard feedback techniques. The pitch response due to a Y input was improved by nearly three times and the roll response due to an X input was almost eliminated showing an improvement of twenty two times.

Introduction

Motion platforms are used in simulators to provide motion cues to the occupant. Realistic cues require the platform to reproduce the desired motion with minimum phase lag, high bandwidth and little cross-talk between degrees of freedom. Late, spurious or ill formed motion cues degrade the effectiveness of the simulation and can result in simulator sickness.[1,2]

In a typical simulator, the desired motion is derived from a vehicle dynamics model using input from the occupant. Because the occupant is in the loop, the motion platform must produce the desired motion with little delay using no look ahead information.

In addition, the platform must be completely safe to the occupant. Motion platforms typically use standard feedback techniques which create error signals between the desired and achieved motion and multiply these signals by gains to create commands for the actuating devices. Higher gains result in increased performance but a lower stability margin. Complex multi-degree of freedom platforms contain highly non-linear dynamics which vary as a function of position. Feedback control loops must be tuned softly, resulting in low bandwidth, in order to ensure stability in all positions.

The control strategy presented here is a technique that combines the computing power of today's processors with detailed nonlinear dynamic knowledge of the motion platform's structural and hydraulic elements. This control system includes a model based feedforward path that determines the required servovalve opening for each actuator to achieve the desired motion. A lightly tuned feedback path runs in parallel, only to keep the motion platform from drifting from the desired position. Because the feedback path is so softly tuned, control stability is not a concern. This control architecture has been successfully implemented on a Stewart platform built for the U.S. Army TACOM TARDEC Physical Simulation Laboratory. This paper presents the basic control architecture and quantitative results.

Physical Description of the Simulator

A Stewart platform built by MTS Systems Corporation for the U. S. Army uses six linear hydraulic actuators to connect a triangular fixed base with a triangular motion platform. The mass of the platform with the payload is 2500 lbm. The six actuators, the base and the platform form a near octahedral structure. Through control of the six actuators, this mechanism provides independent or simultaneous motion of the platform in six natural degrees of freedom. System performance is outlined in Table 1 and a photograph of the system is shown in Figure 1.

Axis	Displacement	Velocity	Acceleration
X (longitudinal)	± 20 in	± 30 in/s	± 1.0 g
Y (lateral)	± 20 in	± 30 in/s	± 1.0 g
Z (vertical)	± 20 in	± 50 in/s	± 2.0 g
Roll (about X)	± 20 °	± 70 °/s	± 1146 °/s^2
Pitch (about Y)	± 20 °	± 70 °/s	± 1146 °/s^2
Yaw (about Z)	± 20 °	± 90 °/s	± 1146 °/s^2

Table 1. Motion Performance of the Stewart Platform

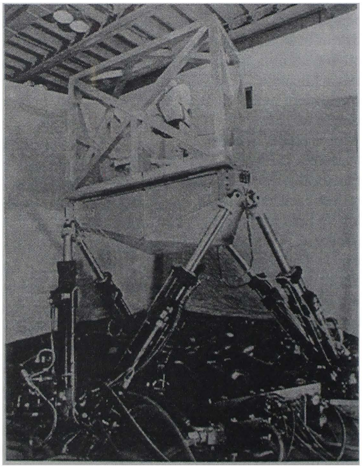


Figure 1. Front View of the Stewart platform built by MTS for the U.S. Army TACOM TARDEC Physical Simulation Laboratory

Control Technique

For many years analog feedback control techniques have been used to control motion platforms for simulation. When using such feedback techniques a basic trade-off between performance and stability arises. Higher performance requires higher loop gains which, in turn, may produce stability problems as the platforms move to different positions. The effective dynamics of a given actuator/mass combination changes when this highly non-linear platform moves to various positions. Low gains must therefore be used to ensure stability at all positions of the platform.

A combination of today's real-time computing power and detailed knowledge of the platform's structural and hydraulic elements have improved motion fidelity without sacrificing stability.

Consider the control system block diagram shown in Figure 2. The structure shown has been found to give excellent results, Hessburg et all (1997)[3] and Clark et all (1998)[4]. It consists of two parts: 1) in-depth platform system knowledge feedforward control and 2) a softly tuned feedback loop to handle modeling imperfections and to keep the system from drifting from the displacement setpoint.

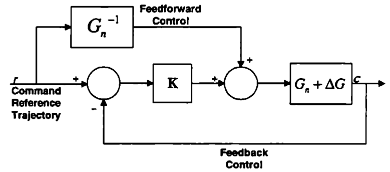


Figure 2. Feedforward with Feedback Control

The feedforward part of the control system uses in-depth, detailed knowledge and accurate modeling of the known payload and hydraulic characteristics. Given the desired motion, the model determines the servovalve opening required for each of the six actuators. These values are further shaped with filters to eliminate excitation of payload resonances. The calculated feedforward servovalve command will achieve the desired platform motion, assuming the *a priori* knowledge is correct. Feedback is used to correct for unknown parameter variations in the total system, and is not required to compensate for the large hydraulic servovalve gain variations and dynamic nonlinearities.

The feedforward process can be described as follows: In Figure 2 **G** denotes the actual plant to be controlled, **G** is broken down into the known part, $\mathbf{G}_n$ (nominal), and the uncertain part, **ΔG**. That is,

$$\mathbf{G} = \mathbf{G_n} + \Delta\mathbf{G}.$$

The term $\mathbf{G}_n$ represents all the knowledge of the "plant" to be controlled: the servovalves, actuators, payload mass and kinematic configuration. These components are described in a mathematical model using nominal parameters that characterize the components. The term **ΔG** represents modeling uncertainty of **G**. This uncertainty may include parameter uncertainties in **G** (e.g., the actual bulk modulus of the oil may be different from the nominal value used in the model) and higher order dynamics. Since $\mathbf{G}_n$ is known, its inverse, $\mathbf{G}_n^{-1}$, can be determined. The term **K** represents the feedback control gain. The feedforward path uses the knowledge of the controlled plant, $\mathbf{G}_n$, and the feedback path, with loop gain factor **K,** compensates for imperfections in the knowledge of the plant, **ΔG**.

The following relationship can be derived from the above block diagram shown in Figure 2, where *r* represents the reference command, and *c* represents the output response of the system:

$$\frac{\mathbf{c}}{\mathbf{r}} = 1 + \frac{\Delta G / G_n}{1 + K(G + \Delta G)}$$

As can be seen by the formula, if the parameter variation **ΔG** is zero, then *c* perfectly follows *r*. When **ΔG** is non-zero, a higher achievable stable gain (**K**) reduces the effect of **ΔG** on the ability of *c* to track *r* (i.e., improves tracking performance).

From this description it is clear that the advance in technology lies in the implementation of $\mathbf{G}_n^{-1}$. MTS chose to implement the controller using an MTS internal software environment called Schema™. Schema™ is a real-time software system that features efficient object-oriented technology, a deterministic process scheduler and connection-based communication. Generic modules were built that accurately model the system's rigid body dynamic and hydraulic characteristics. The only system specific module calculates the kinematic equations of the Stewart platform. Therefore, it is straightforward to implement this control strategy on a wide variety of motion platform configurations.

Figure 3 shows a data flow diagram of the G_n^{-1} implementation. Given the desired motion in an inertial (world) coordinate system, the Inverse Payload Model computes the force required at each actuator. This rigid body model requires the mass properties which can be directly measured using delta pressure or force cells at each actuator.

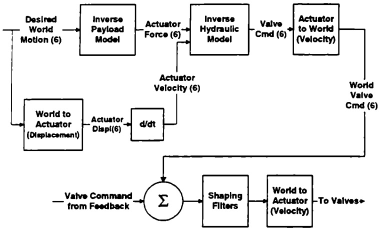


Figure 3. Diagram of G_n^{-1} *Implementation*

The Inverse Hydraulic Model computes the servovalve opening required for each actuator given the desired actuator force and velocity. Effects used in the model include Bernoulli flow through a servovalve orifice, servovalve saturation and oil column compliance. The parameters used in this model, such as the area of the actuator, can be easily obtained from the hydraulic component specifications.

Because the Inverse Payload Model is a rigid body model it has been found that further improvements can be achieved by shaping the servovalve command to reduce the effects of specimen and table flexible dynamics. Because these dynamics do not usually occur in the actuator coordinate system it becomes necessary to transform the servovalve signal back to the world coordinate system. This signal is combined with the feedback signal and shaped to yield a flat closed loop transfer function even in the presence of flexible dynamics.

Setup of the control system is straightforward as the parameters required are simply the physical parameters of the system which can be easily obtained or automatically measured. Examples include the size of the actuators, the flow rating of the servovalve and the mass properties of the payload. Complex tuning of the system, which leaves the operator questioning the overall stability, is not required.

Results

The new feedforward control architecture was installed on the Stewart platform while at MTS and data were collected to determine performance. Excellent results were obtained showing a large bandwidth, low phase lag and low cross-coupling between the degrees of freedom for the entire range of motion. Stability concerns were completely eliminated, as the feedback gains required were extremely low.

Transfer functions were obtained for each degree of freedom and are plotted in Figures 4 - 9. X is in the forward direction, Y is in the lateral direction, Z is down, Roll is the rotation about X, Pitch is the rotation about Y and Yaw is the rotation about Z. The figures show the transfer functions obtained by exciting the system in each degree of freedom with acceleration band limited flat spectrum noise. For the translational degrees of freedom the 1 sigma magnitude was 0.4 g and for the rotational it was 4 rad/s^2 up to 75 Hz. Each plot shows the transfer function at two positions. The solid line represents the transfer function when all actuators were in the center of their stroke or at the world zero position. The dotted line show the transfer function when the simulator was located at the world position as follows:

X	5.0 inches
Y	5.0 inches
Z	6.0 inches
Roll	0.10 radians (5.7 degrees)
Pitch	0.10 radians (5.7 degrees)
Yaw	0.15 radians (8.6 degrees)

For all transfer functions the magnitude in db is shown on the top graph and the phase in degrees on the lower graph. The power of the new feedforward control technique is clearly shown on each transfer function with the flat, wide band magnitude and the small linear phase characteristics. These results show a linear phase in all six degrees of freedom associated with a pure time delay between 9 and 14 ms and less than +/- 2 db magnitude variation to 10 Hz.

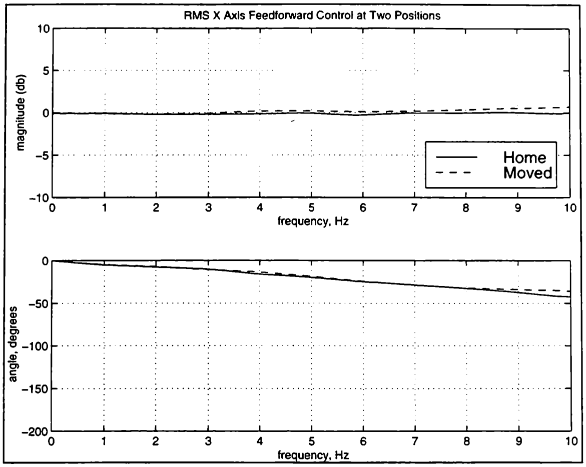


Figure 4. Closed loop transfer functions to X axis excitation at two positions

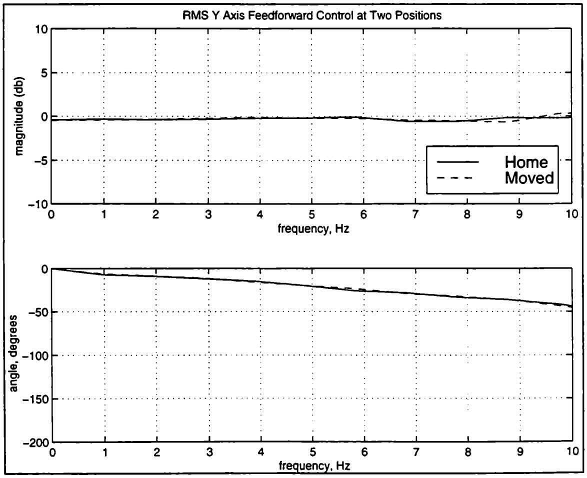


Figure 5. Closed loop transfer functions to Y axis excitation at two positions

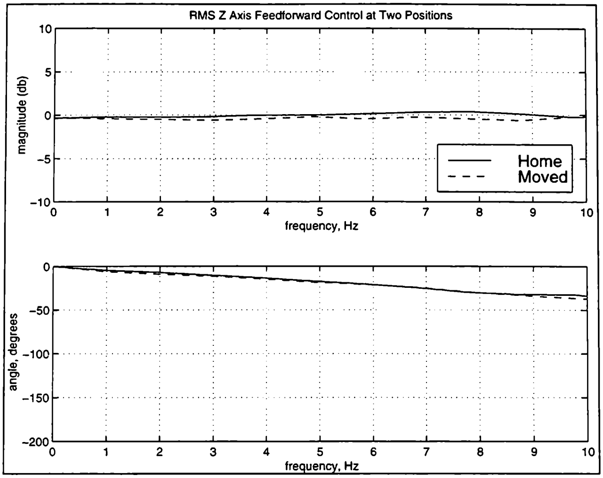


Figure 6. Closed loop transfer functions to Z axis excitation at two positions

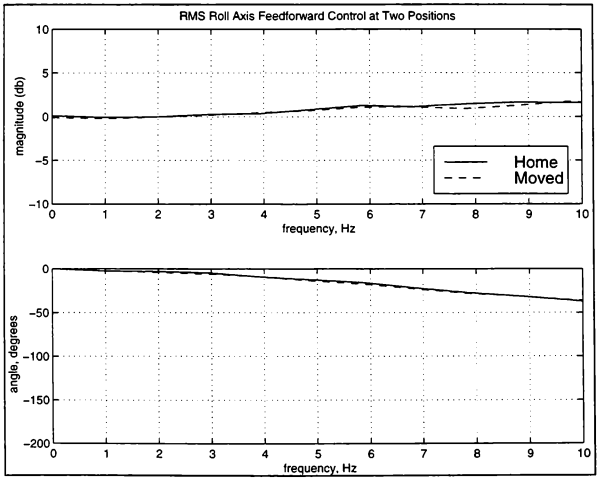


Figure 7. Closed loop transfer functions to Roll axis excitation at two positions

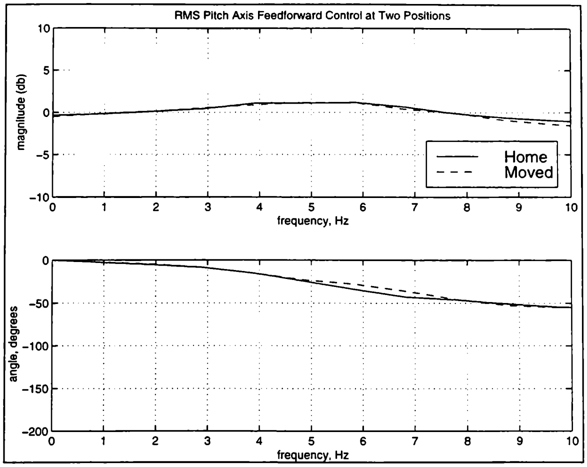


Figure 8. Closed loop transfer functions to Pitch axis excitation at two positions

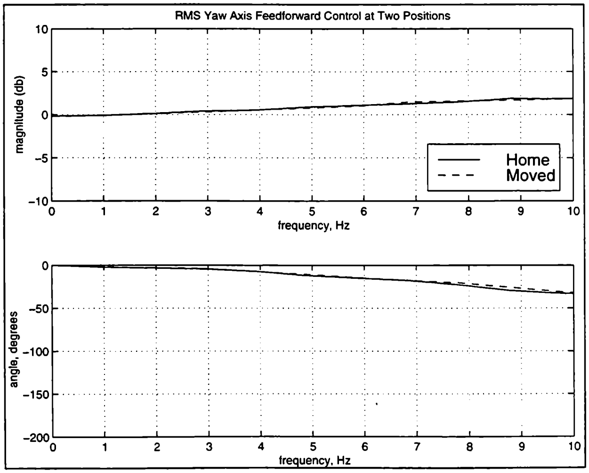


Figure 9. Closed loop transfer functions to Yaw axis excitation at two positions.

Figure 10 shows a time history plot of the desired and actual acceleration of a severe high amplitude, high bandwidth maneuver in the vertical direction.

Figures 11 and 12 show the payload cross coupling disturbance rejection capability of the new control system architecture. Figure 11 shows the transfer function of the roll acceleration response in (radians/s^2) to a random acceleration excitation (g's) signal in the lateral direction. The dashed line shows the amount of cross-talk disturbance of a controller utilizing a well tuned PID with delta pressure compensation. The solid line shows the amount of cross talk with the new feedforward control .. architecture.

Figure 12 shows the transfer functions for the Pitch due to a longitudinal excitation.

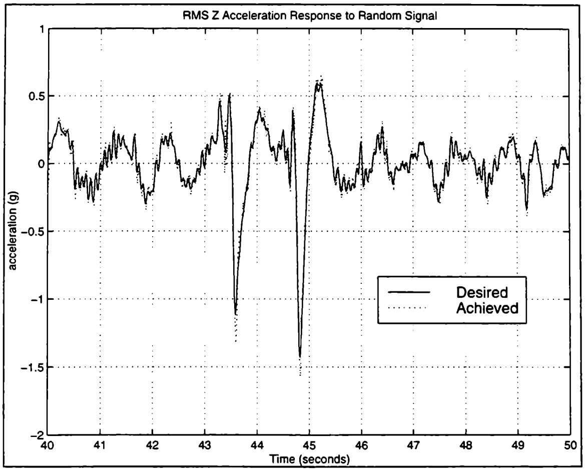


Figure 10. Time history plot of Z direction maneuver

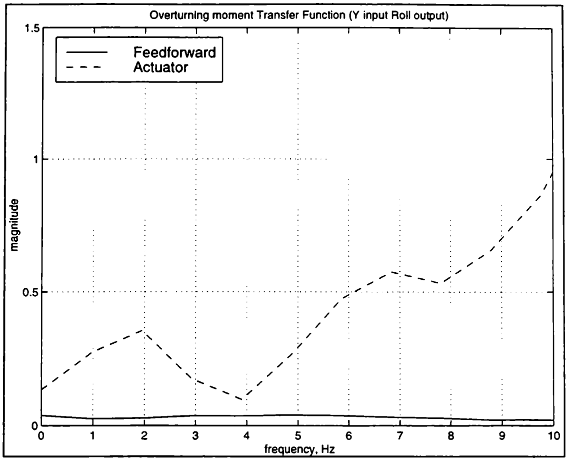


Figure 11. Roll response to Y axis excitation

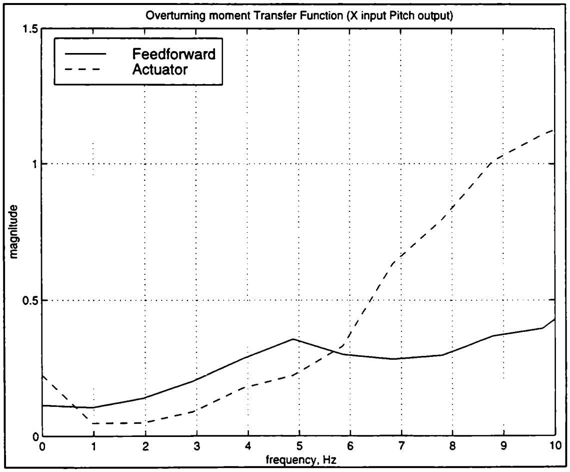


Figure 12. Pitch response to X axis excitation

Because the feedforward control strategy takes into account the high center of gravity it can compensate for the cross-coupling effect between theses axes. From the figures it can be seen that the pitch response due to an X input was improved by nearly three times and the roll response due to a Y input was almost eliminated showing an improvement of twenty two times.

Conclusion

Improved performance of motion platforms used for simulation has been obtained with a sophisticated new control strategy. This control technique gives broad band performance, minimizing phase and cross-coupling without inducing stability problems. Setup of the system is straightforward, as the only parameters required are known physical values. Safety is ensured because only very low feedback gains are necessary to achieve the desired performance.

Acknowledgements

The authors would like to thank the U.S. Army TACOM TARDEC Physical Simulation Laboratory team and the MTS project team, in particular Hugh Sparks, for the work performed on the PSL upgrade project.

References

1. Casali, J.G., "Vehicular Simulation-Induced Sickness, Volume I: An Overview". IEOR Technical Report No. 8501, August 1986.
2. Casali, J.G., and Wierwille W.W., "Vehicular Simulator-Induced Sickness Volume III: Survey of Etiological Factors and Research Facility Requirements". IEOR Technical Report No. 8503, August 1986.
3. Clark A., Larson R., and Thoen B., "Utilizing Modern Digital Signal Processing for Improvement of Large Scale Shaking Table Performance". To be submitted to: Mitigation of the Impact of Impending Earthquakes Conference, International Center for Earthquake Prognostics, October 1998.
4. Hessburg, T., Krantz, D., "Shock Test System Performance Prediction and Feed-Forward Control using High-Fidelity Nonlinear Dynamic Hydraulic System Modeling, Proceedings of the 68th Shock and Vibration Symposium, Baltimore, U.S.A 1997.

AIAA-98-4368

A Low-Cost Flight Simulation for Rapid Handling Qualities Evaluations During Design

Mr. Frederick G. Anderson* and Dr. Daniel J. Biezad†

California Polytechnic State University, San Luis Obispo, California 93407

Abstract

This paper describes the creation of a low-cost, flight simulator and its integration with the AirCraft SyYNThesis (ACSYNT) preliminary design code and with the USAF DATCOM code that estimates stability derivatives. Both ACSYNT and DATCOM are available to universities. The intent of the flight simulator is to allow students to rapidly assess aircraft stability and control parameters and handling qualities metrics for their aircraft designs. This linked package reduces the design cycle time while allowing the student to refine control laws to meet handling qualities constraints while optimizing the aircraft performance. Recent advances in Personal Computer technology have brought workstation-quality performance to the low-cost market. The Cal Poly Flight Simulator takes advantage of these advances by using commercially available, off-the-shelf graphics cards in Pentium computers to produce a highly realistic, three-screen graphical display for the pilot. The pilot flies from the cockpit of an F4/F15 handling qualities simulator cab on loan from NASA Dryden.

Introduction

Current aeronautical research is expanding conceptual design and synthesis capability to incorporate agility and handling quality metrics. This expanded capability will allow designs to be developed that include thrust vectoring, vortical flow modification, and advanced control techniques. This new capability resides primarily in aircraft computer design codes, many of which are available to universities. At Cal Poly, San Luis Obispo, interactive computer design aids are being developed and integrated for rapid design as recommended by an Industrial Advisory Board consisting of mentors from industry and government. The goal is to develop a rapid tool for the evaluation of aircraft stability, control, maneuverability, and handling qualities very early in the design process. To accomplish this goal, the Cal Poly Aero Department has undertaken the Pangloss Project.

The Pangloss project, shown schematically in Fig.1, is an effort to integrate and streamline the estimation and analysis of aircraft dynamics and handling qualities into the senior design curriculum. The overall system consists of three major parts. The first is the design software used by the students to help design aircraft to performance specifications. The AirCraft SYNThesis (or ACSYNT) program is one such tool. It was developed by systems analysis and aircraft design engineers at NASA Ames and is currently administered by Virginia Polytechnic University[1].

* Aeronautical Engineering Master's Student. Staff Engineer: Systems Technology Inc., Hawthorne, California, 90250. Student member, AIAA

† Professor, Department of Aeronautical Engineering, Associate Fellow, AIAA

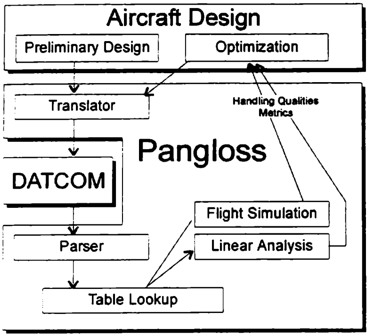


Figure 1: Schematic Diagram of Pangloss Project

The second key feature is stability derivative estimation. Stability and control derivatives in the academic environment are often computed from published tables, curves, and equations. Many of them, however, can be estimated using the batch computer program Digital DATCOM that was developed by the U.S. Air Force in the 1970's.[2] Unfortunately, the program is not interactive, may not apply to unconventional configurations, and does not contain provisions for developing control architectures and feedback control laws. There has therefore been considerable effort to link the design program with Digital DATCOM to automate to process of stability derivative extraction.

The third phase of the aircraft dynamics design cycle involves using these stability derivatives to evaluate vehicle dynamics and handling qualities. There are two principal ways this is accomplished: through traditional linear analysis; and through handling qualities flight simulation. It is the latter that is the focus of this paper. The final challenge of the Pangloss project is to accomplish these goals within the limited budget of the Aeronautical Engineering Department.

Simulator Description

Recent advances in Personal Computer technology have brought SGI workstation-quality performance to the low-cost market. The Cal Poly Flight Simulator takes advantage of these advances by using commercially available, off-the-shelf graphics cards in inexpensive Pentium computers to produce a highly realistic, three-screen graphical display for the pilot. In the primary configuration shown in Fig 2., four PC's are linked together via Ethernet. Three of the computers each process one channel of high-resolution, texture-mapped graphics. The simulation dynamics are calculated by one of the side channel graphics computers. The fourth computer serves as an interface with the analog and discrete inputs and outputs of the cab.

The cab, on loan from NASA Dryden, was originally used as an F-4 Phantom training cockpit and was subsequently used as an F-15 Eagle handling qualities simulator. It is the combination of Phantom and Eagle that gives the simulator the informal name Pheagle. The Pheagle cab is driven by an analog computer called the Siblinc. Reference voltages supplied by the I/O Pentium's digital-to-analog (D/A) converter are amplified by the Siblinc to actuate the analog display gauges in the cockpit. The Siblinc buffers inputs to the I/O computer's analog to digital (A/D) converter from the cab's stick, rudder pedals, throttles, and various discrete switches. The Siblinc also drives a torque-motor force feel system for both axes of stick motion as well as for the rudder pedals. Stick Force parameters such as gain, inertia, and viscous damping can be altered in real time by applying reference voltages from the PC. In this way, the I/O computer can generate force profiles as a function of the aircraft flight state. Furthermore, students can gain valuable insight into the effect of stick shaping on handling qualities.

The simulator can either be run in an interactive, real-time mode or as a constant time-step, batch simulation. In both modes data logging is possible. When run in the real-time mode, the pilot is presented with an out-the-cockpit view, upon which a Head Up Display (HUD) has been superimposed. Control inputs are received either via the network connection during a multi-computer run, or from a joystick when the program is running stand-alone. These inputs are then filtered through the control module that allows the user to include a feedback control system in the simulation. Multiple control architectures can be switched between as the simulator runs. In batch mode, control inputs are

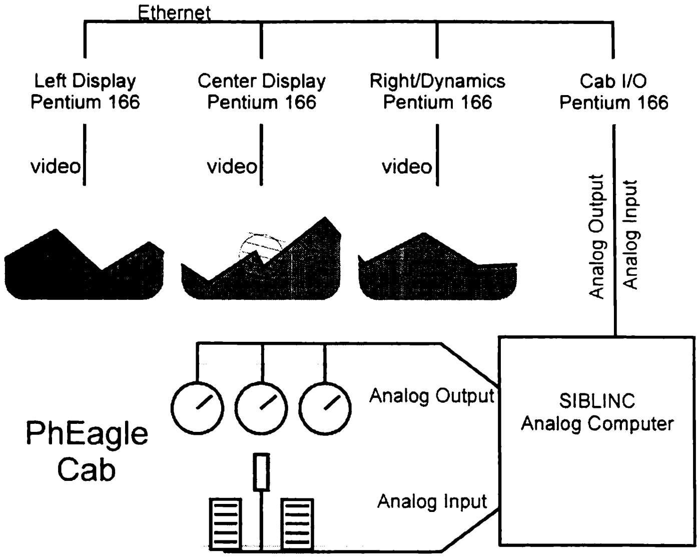


Figure 2: Configuration of the Multi-computer Simulation

defined by the user prior to run-time and executed by the computer during the course of the run.

The heart (or engine!) of a flight simulator is its set of equations of motion. Pheagle is capable of operating either as a discrete simulation of a continuous-time linear system, or as a full six-degree of freedom (6DOF) non-linear simulator. The linear simulation interfaces with Engineering analysis tools such as Matlab and Program CC. This makes it especially useful for design students to rapidly investigate the effect of changing control laws plant parameters. In this mode of operation, the user creates a transfer function system that the simulator reads as a state-space quadruple. The matrix elements, A, B, C, and D, of this quadruple define the parameters of the state equation

$$\begin{aligned}\dot{\mathbf{X}}(t) &= \mathbf{AX}(t) + \mathbf{BU}(t) \\ \mathbf{Y}(t) &= \mathbf{CX}(t) + \mathbf{DU}(t)\end{aligned} \tag{1}$$

Where **X**(t) is the system state vector, **U**(t) is the input vector, and **Y**(t) is the output vector. Assuming time-invariant coefficients and an Euler integration scheme, the discrete-time system:

$$\begin{aligned}\mathbf{X}(k+1) &= \mathbf{GX}(k) + \mathbf{HU}(k) \\ \mathbf{Y}(k) &= \mathbf{CX}(k) + \mathbf{DU}(k)\end{aligned} \tag{2}$$

matrices G and H can be solved for by

$$\begin{aligned}\mathbf{G} &= \mathbf{I} + \mathbf{A}dt \\ \mathbf{H} &= \mathbf{B}dt\end{aligned} \tag{3}$$

where dt is the simulation sample rate.

In the full 6DOF mode of operation, forces and moments are generated via stability and control derivatives. These forces and moments are then converted into the aircraft body axis and used to calculate both linear and angular acceleration.

Assuming constant mass, the equation of linear motion can be expressed[3]

$$\mathbf{F} = m\frac{d\mathbf{V}}{dt} + m\Omega \times \mathbf{V} \tag{4}$$

Where F is the force vector [Fx, Fy, Fz], **V** is the linear velocity vector [u, v, w], and Ω is the angular velocity in the body frame [p, q, r]. Since the force vector is known, this equation can be solved for dV/dt:

$$\frac{d\mathbf{V}}{dt} = \frac{\mathbf{F}}{m} - \Omega \times \mathbf{V} \tag{5}$$

The resultant vector dV/dt is then numerically integrated to become the aircraft velocity in the body frame. This is transformed via a direction cosine matrix into flat-earth, inertial velocity [vx, vy, vz]. Finally, this velocity is integrated to yield inertial position.

The equation of angular motion is expressed:

$$\mathbf{M} = \frac{d\mathbf{H}}{dt} + \Omega \times \mathbf{H} \tag{6}$$

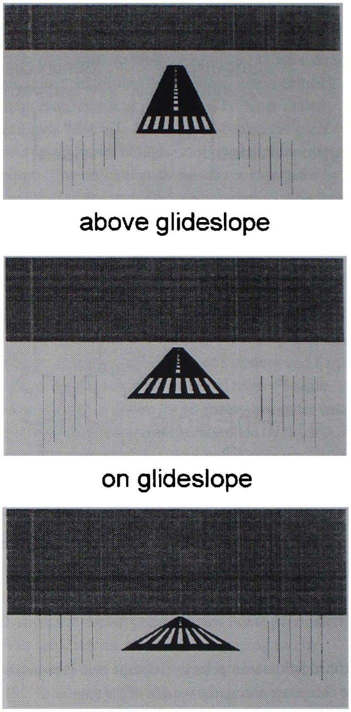
where M is the moment vector [l, m, n], Ω is the angular velocity in the body frame [p, q, r], and **H** is angular momentum, **I**·Ω. Assuming a constant moment of inertia matrix, **I**(dΩ/dt) can be substituted for **dH**/dt. dΩ/dt is then given as:

$$\frac{d\Omega}{dt} = \mathbf{I}^{-1}\mathbf{M} - \Omega \times \mathbf{H} \tag{7}$$

The angular acceleration is integrated and transformed to Euler rates that are integrated into the inertial heading, pitch, and roll angles Ψ, Θ, and Φ.

Handling Qualities Evaluation Features

A variety of evaluation techniques have been developed in order for the designer/pilot to quantitatively assess the handling qualities of the aircraft. In the real-time simulation mode, either a landing task or an "up-and-away" flying task can be used. In the landing task, the simulation is begun with the aircraft displaced both horizontally and vertically from the glideslope to the runway. The pilot's task is to intercept the glideslope and fly a final approach, flare and touchdown. A score for the approach is assessed based upon the mean square of the aircraft's linear deviation from the glidepath. The landing is scored based upon vertical speed at touchdown, distance from the runway's centerline, and distance from the touchdown point.



above glideslope

on glideslope

below glideslope

Figure 3: "Telephone Pole" Glideslope Indicators

In order for the pilot to determine the proper glideslope, a series of "telephone poles" have been placed at the end of the runway. The height and placement of these poles is such that when the pilot is on glideslope, the tips of the poles appear even with the horizon. When viewed from above or below, the tips of the poles form a peak or valley, respectively (Fig 3).

The landing task effectively evaluates the handling qualities of the aircraft over a large operational

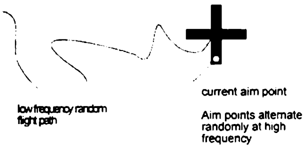

current aim point
Aim points alternate randomly at high frequency
low frequency random flight path

Figure 4: Up and Away Target Task

frequency band. Maintaining the glideslope at long range is a relatively low bandwidth task, while doing so at close range requires higher frequency control. Atmospheric turbulence is added to prevent the pilot from learning to perform the task "open loop".

The "up-and-away" task requires the pilot to track a moving target. This cross-shaped target traces a path across the horizon while 4 aiming points at the tips of the cross alternate randomly (Fig. 4). The trajectory of the target is controlled independently in the vertical and horizontal directions by sum-of-sines functions. The aiming points alternate randomly within a frequency band of 0.5 to 3.0 rad/sec. Scoring is based upon the mean square angular error in aiming.

Naturally, the scores assigned during the landing and pointing tasks are dependent upon the individual skill of the pilot and therefore highly subjective. When used to as a comparison between two aircraft with a single pilot, however, they can help identify handling qualities differences between the two configurations.

Visual Scene

Graphics technology for the Personal Computer has seen a period of accelerated growth over the past few years. It is now possible for an inexpensive computer to achieve rendering performance previously available only on high-priced workstations. The result of this development is that high quality simulation graphics is no longer available only to large engineering corporations and government agencies. Since this technological advance is being driven primarily by widespread consumer demand, the latest versions of popular PC games are likely to have graphics comparable to those of a multi-million dollar simulator of a few years ago.

The Cal Poly Simulator is taking advantage of these recent developments. The three display computers contain low cost graphics accelerator cards made by Quantum3D based on the 3Dfx Voodoo chip[4]. Like several of the current generation of graphics hardware accelerators, this card provides hardware texture-mapping, depth buffering, and Gouraud shading. It has a pixel fill rate of over 90 Mpixels/second at 800 x 600 resolution and can render a scene containing upwards of 5 to 10 thousand polygons at 30 Hz.

In order to keep development time to a minimum, the Pheagle simulator uses an off-the-shelf, real-time graphics rendering software package. Gemini Technology donated a site-license for their OpenGVS product to help with the Cal Poly simulator effort. This rendering library is linked with the real-time code to handle the details of supporting multiple model file formats, scene database management, lighting and atmospheric effects, as well as object and terrain collision detection.

Conclusions

Cal Poly's Pheagle handling qualities flight simulator gives design students rapid, meaningful feedback about how their design decisions effect the handling qualities of their aircraft. By using commercially available, off-the-shelf graphics cards in inexpensive Pentium computers, Pheagle, provides high-quality simulation at a low cost. As part of the larger Pangloss project, this simulator is helping to integrate and streamline the estimation and analysis of aircraft dynamics and handling qualities into the senior design curriculum.

References

[1] Myklebust, A., and P. Gelhausen, "Putting the ACSYNT on Aircraft Design," Aerospace America, Vol. 32, No. 9, Sept. 1994 pp. 26-30.
[2] The USAF Stability and Control Digital DATCOM, Volume II, "Implementation of Datcom Methods," AFFDL-TR-79-3032, McDonnell Douglas Company, St. Louis, Missouri, April 1979

[3] McRuer, D., Ashkenas, I., and Graham D., Aircraft Dynamics and Automatic Control, Princeton University Press, Princeton, New Jersey, 1973

[4] "Survey: Low-Cost 3-D Graphics Accelerators," Real Time Graphics Newsletter, published by Computer Grâphics Systems Development Corp. vol. 5 no. 9, April/May 1997

AIAA-98-4369

LOW-COST SIMULATION AND MILITARY AVIATION TEAM TRAINING

Renée J. Stout, Ph.D.
Eduardo Salas, Ph.D.
Danielle C. Merket
Naval Air Warfare Center Training Systems Division*
Orlando, FL
Clint A. Bowers, Ph.D.
University of Central Florida*
Orlando, FL

Many modern job environments impose the need for teams of individuals to work together to effectively and efficiently accomplish their tasks, due to the complex nature of these tasks. Rapid changes in the workforce and workplace, such as advancements in technology and decreases in resources, suggest that teams will continue to be used to perform a variety of functions well into the future. In these settings, proficient task accomplishment depends upon teams employing appropriate and coordinated teamwork processes[1,2]. This element of teamwork is paramount in that ineffective interaction among team members can cause severe performance effects. For some teams, such as military teams, surgical teams, and aircrews, potentially catastrophic results may occur when teamwork processes breakdown[3]. For example, there is considerable documentation of dire consequences resulting from ineffective teamwork, including mishaps attributed to breakdowns in skills such as communication, leadership, and decision making[4,5,6,7].

The need to train these teams to coordinate their activities is obvious[8]. This is especially apparent in military aviation given the dynamic characteristic of cockpit conditions and the potentially extreme consequences of ineffective teamwork in this arena[3,9]. The key to successfully training teamwork in complex team settings is to delineate the requisite team Knowledge, Skills, and Attitudes (KSAs) for effective mission accomplishment and to provide information, demonstration, practice and feedback to build these KSAs in context[10,11]. Task simulation has been offered as an effective way to train necessary KSAs[10,12]. Within the commercial airline industry the use of Line-Oriented Flight Training (LOFT), via simulation that holds high physical fidelity, to train Crew Resource Management (CRM) has become commonplace. Unfortunately, high-fidelity simulation is extremely costly. As a result, researchers have argued that the use of innovative technologies, such as low-cost simulation, can offer a productive means of delivering training on necessary teamwork KSAs for attaining task success in complex team areas like military aviation[10,13,14,15].

Low-cost simulation offers several advantages in addition to economy. While not being highly realistic in physical characteristics, scenarios designed for low-cost simulation can be specifically designed to elicit desired KSAs thus maintaining functional fidelity. Also, low-cost simulation can serve as a springboard for more advanced training methods and strategies. Prince and Salas[15] argued that more complex training, such as LOFT, should come after the trainees have developed more basic KSAs. Additionally, low-cost simulation provides flexibility of use in that it is transportable and thus available in a variety of settings. This can therefore provide aviators with multiple opportunities to practice and to receive feedback on teamwork skills.

Given the strong potential that low-cost simulation can provide for training teamwork KSAs, in this paper, we discuss a set of lessons learned from our own extensive body of research in designing, developing, and delivering training for naval aviators to enhance their teamwork skills using low-cost simulation. Our experiences have ranged from using low-cost, off-the-shelf software, and PC-based systems for use in demonstrations of teamwork, roleplay exercises, and carefully crafted scenarios geared toward developing teamwork skills. We have applied our systematic methodology for improving teamwork

* The views expressed in this article are those of the authors and do not represent the official positions of the organizations with which they are affiliated.

coupled with the employment of these low-cost flight simulations at all levels of naval aviation, including undergraduate training, fleet replacement squadron training, and fleet training. We will offer concrete examples of how we have applied low-cost simulation in the research investigations that we have conducted, as well as for training teamwork skills in aviation environments. We will also present data which suggests that our training has been successful. Finally, we will discuss the implications that our research has for scenario design using low-cost systems and will provide guidance for conducting aviation team training using low-cost flight simulations.

We first turn to describing research that we have conducted investigating team performance using low-cost simulation. Much of this research was conducted under a program on CRM training for military aviation, and as such looked at ways that low-cost simulation can support the development of CRM skills. These skills for the Navy were developed through an extensive coordination demands analysis and were described by Prince and Salas[16]. These skills are: decision making, assertiveness, mission analysis, communication, leadership, adaptability/flexibility, and situational awareness.[†]

<u>The Use of Low-Cost Simulation for Research Purposes in the Laboratory</u>

We have conducted several studies using the basic paradigm of low-cost simulation illustrated in Figure 1 to investigate team training and performance, and will summarize a few below. This laboratory design for team performance research was described in further detail by Bowers et al.[14]. When employing undergraduate university students as research participants, task simulations used have included "Gunship[17]", "Falcon[18]", and "Comanche[19]". Thus, scenarios or missions performed using these simulations included surveillance, air-to-air combat, and air-to-ground combat. In essence, with this paradigm, interdependence among team members is created by dividing tasks, such as allocating keyboard functions to a "co-pilot" to control altitude and targeting functions, and a joystick to a "co-pilot" to control speed and heading functions. The use of peripherals, such as a video-splitter to simultaneously display information to each crew member, partitions to encourage verbal communication, headsets to create realism, and recording devices to capture verbal and non-verbal behaviors are also utilized.

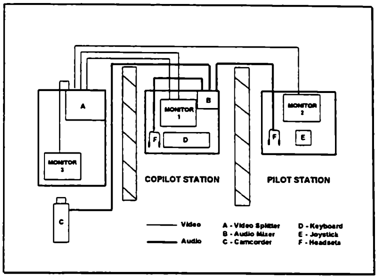


Figure 1. Schematic illustration of the low-cost simulation. <u>Note.</u> Adapted from Bowers, Salas, Prince, & Brannick (1992).

Using this basic design and the Gunship simulation, Lassiter, Vaughn, Smaltz, Morgan, & Salas[20] sought to determine the effect on team communications (i.e., one of the CRM skills identified by Prince & Salas[16]) and performance of two different training interventions. The control group received no training. According to these authors, training was either knowledge-based (i.e., a lecture on important concepts related to effective communication) or skill-based (i.e., a video demonstration modeling effective communication skills). Results indicated that participants who received the behavioral modeling training used better quality communications during the low-cost simulation mission than either those receiving the lecture or the control group. Further, while training did not lead to greater performance, those teams that communicated better performed better, indicating that effective communication is important to team performance.

Using the Gunship simulation, Smith & Salas[21] investigated the effect of three different training manipulations as compared to no training on another CRM skill identified by Prince and Salas[16] -- assertiveness. Training was either one hour of lecture,

[†] While we use the term skills training here, our CRM training also imparted knowledge and attitudes thus covering the gamut of teamwork KSAs.

lecture with behavioral modeling, or lecture with behavioral modeling and active roleplay with feedback. Assertiveness on the part of the participant in response to actions of a confederate, who created five conflict situations during a low-cost simulation, was recorded using a behaviorally anchored rating scale. Results indicated that participants who received training involving active practice via role play were found to demonstrate better task assertiveness during the low-cost simulation. Neither of the other two training interventions were found to impact assertive behavior. This provides support that active practice is important to improving CRM skills, such as assertiveness.

Brannick, Roach, and Salas[22] assessed the relationship between teamwork and performance in the Falcon simulation. Two measures of teamwork processes were used. One was a communication content analysis considering simple frequencies of overt communications engaged in by the team. The other used a rating scale to determine the quality of a subset of teamwork skills identified by Morgan, Glickman, Woodard, Blaiwes, and Salas[23] which served as one of many sources used in the identification of the Prince and Salas[16] CRM skills. Performance was assessed via two outcome measures provided by the simulation itself (i.e., time to down the enemy and number of radar locks obtained by the enemy on the team's craft) and one subjective measure (i.e., in one case, an on-site observer judgment of the quality of overall performance during the task). Results indicated that teams rated higher in quality on the teamwork skills also performed better according to the subjective performance measure, while communication frequencies were not generally highly correlated with this performance measure. Results also suggested that, in general, both teamwork measures were associated with the objective outcomes, lending further support to the importance of teamwork and performance.

Using the Gunship simulation, Stout, Salas, & Carson[24] investigated whether teams that used better CRM skills (according to Prince & Salas[16]) exhibited better performance over that contributed by their individual task proficiency (i.e., skills in performing keyboard and joystick functions). Performance of the team was assessed by a subjective rating of overall mission effectiveness by a task expert and by the number of targets destroyed by the team. Multiple regression analyses revealed that better teamwork was related to mission effectiveness when individual task proficiency was held constant. Evidence was also produced that teamwork exhibited by the "pilot" and team as a whole was significantly related to the number of targets destroyed. In addition, results suggested that there was a differential effect on performance depending upon the specific measure used and whether or not "pilot" or "copilot" teamwork was being investigated. This suggests that further work should be conducted to look at the relationship of teamwork skills needed by each interdependent member of the team.

Taken as a whole, these studies using low-cost simulation in the laboratory provide important implications regarding the role of teamwork and performance. Driskell and Salas[25] argued that results, such as those obtained, have a high probability of transferring to more operational populations. Therefore, we were able to use these results as a starting point for structuring training for aviation teams. Of course, to increase their meaningfulness to operational environments, we also conducted specific low-cost simulation research using aviators as participants as described next.

The Use of Low-Cost Simulation for Research Purposes in the Field

The basic simulation that we have used for conducting research with aviators is similar to that illustrated in Figure 1. In this research, we have utilized the low-cost simulation, Microsoft Flight Simulator[26]. We have included various peripherals, such as yokes for communities who use these and joy-sticks for others, standard approach plates, maps, briefing materials, and checklists. All of this helps to make what would otherwise be a computer game a somewhat more realistic simulation. However, nothing is more important in accomplishing this goal than carefully crafting the scenario around which the simulation is organized. For both research purposes and specific training applications with which we have been involved employing low-cost simulation, we have used an Event-Based Approach to Training (EBAT) that we have also applied to full mission simulation. The EBAT methodology is depicted in Figure 2.

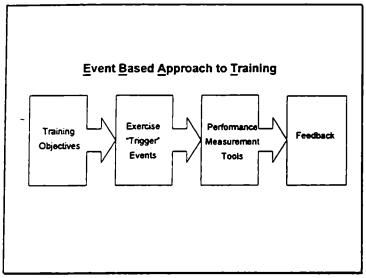

Event Based Approach to Training
Training Objectives
Exercise "Trigger" Events
Performance Measurement Tools
Feedback

Figure 2. The EBAT methodology.

With this methodology, training needs are identified and objectives are specified based upon these needs. The objectives then serve to guide the building of specific events into the scenario in an attempt to elicit the required KSAs from the trainees. Thus the events serve as stimulus triggers for the crews to engage in specific desired behaviors. Extreme care is taken to ensure that scenario events are as realistic as possible while still imparting the requisite training. Performance measurement and feedback tools are linked to these specific scenario events, allowing recording of specific behaviors and allowing targeted feedback to the crews on their demonstration or lack of demonstration of critical team KSAs. Through this iterative process, diagnosis of performance deficiencies then serves to tailor subsequent training.

One example of our use of the EBAT methodology involved work conducted with the primary undergraduate training community. In this case, a thorough needs analysis revealed that assertiveness was an important skill in need of training in this community. Therefore, we created a scenario in which the instructor provided cues to the trainee that he was experiencing vertigo as a result of food poisoning, and he eventually flew the aircraft into the clouds. Around this particular event in the scenario, we were looking for a set of specific behaviors on the part of the student, including his/her taking control of the aircraft when it became necessary to do so. Perhaps not surprisingly, over 90% of the students allowed the instructor to go into the clouds without taking control of the aircraft and thus were provided with feedback as to the error in their behavior. This illustrated the potential utility of the EBAT approach for improving critical team KSAs. Subsequent studies and applications described here also used this EBAT approach.

Prince, Brannick, Prince, and Salas[27] described low-cost simulation research that was conducted with a military undergraduate aviation community to determine the construct validity of a behavioral assessment scale that was developed to rate the seven skills of CRM[16]. Participants were sixty-nine students and thirty-three instructors that performed the scenario in teams of two. These researchers offered evidence that CRM is multi-dimensional and distinguishable by behavioral assessors. Also, this research importantly pointed out the need for more careful attention to scenario design. That is, while pilot subject matter experts had designed two scenarios for this research that were intended to be of equivalent difficulty, performance data indicated that the second scenario was of much greater difficulty. This indicates the criticality of punctiliously pilot testing scenarios whenever possible. In addition, Baker et al.[13] presented reaction data obtained in this investigation which indicated that both student and instructor pilots perceived the low-cost simulation to be an acceptable means of training CRM skills.

A more recent study was conducted to investigate the transfer effects of training via low-cost simulation[28]. Participants were forty-eight teams of two from a military undergraduate helicopter community who had completed their formal training and were awaiting assignment to their respective fleet replacement squadrons (e.g., CH-46, SH-60). Half of the teams had received training by performing together in a low-cost simulation scenario, while half had worked together for an equivalent time period performing generic team building exercises. Results indicated positive transfer of skills from the low-cost simulation to the high-fidelity simulation on two specific items that were emphasized during the low-cost simulation, suggesting the value of low-cost simulation to developing these types of skills.

The research that we have conducted within military fleet settings has provided support through both reaction and performance data that low-cost simulation can yield training value. We next turn to describing some uses of low-cost simulation for specific training applications in the field, as well as some anecdotal evidence that each specific training application was effective.

<u>Low-Cost Simulation to Support Training Via Roleplay Exercises</u>

One example of our use of low-cost simulation to provide aviators with training involved utilizing the simulation to support a communication training roleplay exercise with a military undergraduate community. The training was formally implemented in their CRM class. Prior to conducting the course communication training module, which involved lecture, videotape presentations, and case study analysis, two students were selected from the course to perform the roleplay exercise and the other students recorded behaviors that they observed on a checklist. In this roleplay, the students were flying a cross country mission when they encountered a Search and Rescue (SAR) aircraft which was trying to locate a downed Cessna (both roleplayed by the instructors). As the roleplay unfolded, it became evident that the SAR and Cessna could not intercept because they were talking on two different radios (i.e., one on UHF and one on VHF). An effective communication strategy would have been to divide the tasks such that the students each took one radio and efficiently relayed the information back and forth to help the SAR locate the Cessna. Anecdotal observation of four classes revealed that students performed fairly poorly during this roleplay. They were then given feedback on how to use better communication skills to more effectively handle the situation while the other students in the class observed. Next all students were presented with the communication module. Finally, after the formal training, two other students performed the exact same roleplay. Anecdotally, due to the training, their communication skills had greatly improved. However, we presented the pilot flying with a cue card telling him to "gradually decrease altitude and fly the aircraft into the ground". Indeed, as in the EAL 410 crash into the Florida Everglades, the copilot had become so occupied with communicating that aviating failed and they flew the aircraft into the ground. This happened in all four courses observed, and students reported that this was a highly effective training experience.

In addition to roleplay training applications, we have used the EBAT methodology to develop specific training that was applied to aviation communities using low-cost simulation. We next describe two such applications.

Training Applications Using Low-Cost Simulation

One example of our use of low-cost simulation in conjunction with our EBAT methodology was applied to a military helicopter aviation operational community as part of their formal CRM training course. This community lacked full mission simulators and thus used low-cost simulation as an alternative to provide practice on critical team skills. In addition, through a research initiative, we developed a low-cost simulation for members of the helicopter crew who are the mechanics in the back of the aircraft (i.e., crew chiefs). Thus the pilot and copilot flew Microsoft Flight Simulator, adapted to more closely model helicopter parameters, while the crew chief operated the other low-cost simulation, which showed digitized images of the back of the aircraft with displayed messages of particular aircraft problems. The scenarios required them to work together as a crew of three. We helped the instructors of this course to develop the three different CRM scenarios used during the course. While a crew performed a given scenario, the remainder of the class recorded CRM behaviors observed during the flight and observed the feedback session following the flight. Anecdotal evidence presented by the instructors indicated that the third crew always performed much better than the second crew, who in turn performed better than the first. This provides some evidence that observing CRM performance and feedback sessions enabled the trainees to learn CRM skills.

Finally, we are currently working on a research project that will provide specific training applications as products. In this effort, we are building low-cost simulation software to model three undergraduate training aircraft - two fixed wing and one helicopter. The modeling of the helicopter represents a significant software challenge. In addition, we are incorporating our EBAT methodology (which has traditionally been paper-based) via support tools designed into the system, making the tools available to an instructor through a menu-driven system. This will help the instructor to design new scenarios based upon emerging training needs, as well as to develop performance measurement and feedback tools that are linked to this scenario. Each of the three undergraduate aviation communities have plans to use the simulations during their CRM training.

Thus, taken as a whole, the examples of our use of low-cost simulation provided in this paper provide evidence of its utility to study CRM processes from which training can be derived, as well as both anecdotal and empirical evidence of the effectiveness of the training that is delivered. Given the various experiences that we have had using low-cost simulation, we have gathered a number of lessons learned and guidelines for employing this type of simulation for both research and training purposes. We will conclude this paper by highlighting a few of these.

Guidance for Using Low-Cost Simulation

1. *Nothing is More Important than the Scenario*. Without a realistic, carefully crafted scenario, the simulation can be perceived as nothing more than a

computer game. The same time and attention should be taken to create scenarios for use with low-cost simulation as would be done with any other type of simulation. With most commercially available off-the-shelf software packages for use with PC-Based systems, simulations of aircraft specific emergencies is not available, nor is it necessarily desirable. Care should be taken to focus on embedding events that require teamwork on the part of the crew rather than procedures training.

2. *Extensively "Pilot-Test" the Scenario.* There is nearly as much art to effective scenario development as there is science. Even when heavily involving subject matter experts in the scenario design process, care must be taken to ensure that necessary KSAs are being tapped and to ensure that control is maintained in the scenario, to make it a valuable learning event, without sacrificing realism. This can only be accomplished by observing crews performing the scenario and iteratively modifying the scenario as needed to maintain consistency.

3. *Describe Capabilities and Limitations of the Simulation Up Front.* It should be pointed out that the simulation will not necessarily feel or act like the real aircraft. It should be emphasized that the purpose of using the simulation is to focus on decision making and teamwork skills so that participants/trainees are not unduly occupied with the handling of the system.

4. *Provide Individuals with Practice on Using the System.* Individuals should be provided with time to get used to the workings of the low-cost simulation. When using aviators and Microsoft Flight Simulator, we have found that 15 minutes of practice is sufficient. When using undergraduate university students in the laboratory, considerably more time is required to attain task proficiency.

5. *Use Relevant Peripherals Whenever Possible.* Headsets, sound boards, and relevant flight controls increase the realism of the simulation as do the use of approach plates, checklists, and navigation aids. By engaging the participants, these peripherals help to increase their motivation.

6. *Provide a System to Record CRM Behaviors.* VHS cameras, video-capture cards, and stereo-audio all help in capturing the crews performance. This is useful for feedback as well as evaluation purposes.

7. *Use Low-Cost Simulation as an Alternative to Full Mission Simulation but Never as a Replacement of this Latter Training.* There are several skills that can be developed through using low-cost simulation training, but other skills require higher fidelity simulation for effective learning. Low-cost simulation can be used to prepare trainees to perform in higher fidelity environments.

Summary

We have provided several examples of uses of low-cost simulation in laboratory environments that have applicability to generating training strategies that we have in turn tested in operational military aviation environments. In addition, we have discussed some of the training applications in which we have been involved. We described our EBAT methodology, which is the cornerstone to making low-cost simulation, as well as other types of simulation to be used, valuable learning tools. Finally, we listed a few points to consider when using low-cost simulation to study and train team skills. We hope that this information is useful to researchers and practitioners seeking low-cost but effective training methods for use in operational team environments. The innovative work of Neil Johnston of Aer Lingus provides one example of a successful transition of our methodology to a different operational community. We hope that others will follow.

References

1 Salas, E., & Cannon-Bowers, J. A. (1997). Methods, tools, and strategies for team training. In M. A. Quiñones & A. Ehrenstein (Eds.), Training for a rapidly changing workplace: Applications of psychological research (pp. 249-279). Washington, D. C.: APA Press.

2 Salas, E., Dickinson, T. L., Converse, S. A., & Tannenbaum, S. I. (1992). Toward an understanding of team performance and training. In R. W. Swezey & E. Salas (Eds.), Teams: Their training and performance (pp. 3-29). Norwood, NJ: Ablex.

3 Cannon-Bowers, J. A., Salas, E., & Converse, S. A. (1993). Shared mental models in expert team decision making. In N. J. Castellan, Jr. (Ed.), Individual and group decision making: Current issues (pp. 221-246). Hillsdale, NJ: LEA.

4 Cooper, G. E., White, M. D., & Lauber, J. K. (Eds.). (1980). Resource management on the flight deck: Proceedings of a NASA/Industry workshop (NASA CP-2120). Moffett Field, CA: National Aeronautics and Space Administration, Ames Research Center.

5 Helmreich, R. L., & Foushee, H. C. (1993). Why crew resource management? Empirical and theoretical bases of human factors training in aviation. In E. L. Weiner, B. G. Kanki, & R. L. Helmreich (Eds.), Cockpit resource management (pp. 3-45). San Diego, CA: Academic.

6 Lauber, J. K. (1987). Cockpit resource management: Background studies and rationale. In H. W. Orlady & H. C. Foushee (Eds.), Cockpit resource management training: Proceedings of the NASA/MAC workshop (NASA CP-2455, pp. 5-14). Moffett Field, CA: National Aeronautics and Space Administration, Ames Research Center.

7 NTSB (anon.). (1994). Safety study: A review of flightcrew-involved, major accidents of U.S. carriers, 178 through 1990 (NTSB/SS-94/01). Washington, DC: NTIS.

8 Goldstein, I. L. (1993). Training in organizations: needs assessment, development, and evaluation. Pacific Grove, CA: Brooks/Cole.

9 Stout, R. J., Salas, E., & Kraiger, K. (1997). The role of trainee knowledge structures in aviation team environments. The International Journal of Aviation Psychology, 7, 235-250.

10 Cannon-Bowers, J. A., Tannenbaum, S. I., Salas, E., & Volpe, C. E. (1995). Defining team competencies and establishing team training requirements. In. R. Guzzo & E. Salas (Eds.), Team effectiveness and decision making in organizations (pp. 333-380). San Francisco, CA: Jossey Bass.

11 Stout, R. J., Salas, E., & Fowlkes, J. (1997). Enhancing teamwork in complex environments through team training. Group Dynamics: Theory, Research, & Practice, 1, 169-182.

12 Jacobs, J. W., Prince, C., Hays, R. T., & Salas, E. (1990). A meta-analysis of the flight simulator training research (NAVTRASYSCEN Technical Report TR-89-006). Orlando, FL: Naval Training Systems Center.

13 Baker, D. P., Prince, C., Shrestha, L., Oser, R., & Salas, E. (1993). Aviation computer games for crew resource management training. The International Journal of Aviation Psychology, 3, 143-156.

14 Bowers, C. A., Salas, E., Prince, C., & Brannick, M. T. (1992). Games teams play: A methodology for investigating team coordination and performance. Behavior Research Methods, Instruments, and Computers, 24, 503-506.

15 Prince, C., & Salas, E. (1991, April). The utility of low fidelity simulation for training aircrew coordination skills. Proceedings of the International Training Equipment Conference, Masstricht, Netherlands, 87-91.

16 Prince, C., & Salas, E. (1993). Training and research for teamwork in the military aircrew. In E. L. Wiener, B. G. Kanki, & R. L. Helmreich (Eds.), Cockpit resource management, (pp. 337-366). Orlando, FL: Academic Press.

17 Micropose Simulation Software (1986). Gunship: The attack helicopter simulation game (Computer Program). Hunt Valley, MD: Micropose Software, Inc.

18 Spectrum Holobyte (1988). Falcon (Computer Program). Spectrum Holobyte, Inc.

19 NovaLogic (1995). Comanche 2.0 (Computer Program). Novalogic, Inc.

20 Lassiter, D. L., Vaughn, J. S., Smaltz, V. E., Morgan, B. B., Jr. & Salas, E. (1990). A comparison of two types of training interventions on team communication performance. Proceedings of the Human Factors Society 34th Annual Meeting, Santa Monica, CA, 1372-1376.

21 Smith, K. A., & Salas, E. (1991, March). Training assertiveness: The importance of active participation. Paper presented at the 37th Annual Meeting of the Southeastern Psychological Association, New Orleans, LA.

22 Brannick, M., Roach, R., & Salas, E. (1993). Understanding team performance: A multimethod study. Human Performance, 6, 287-308.

23 Morgan, B. B., Jr., Glickman, A. S., Woddard, E. A., Blaiwes, A. S., & Salas, E. (1986). Measurement of team behavior in a Navy environment. (NTSC Tech. Rep. No. TR-86-014). Orlando, FL: Naval Training Systems Center.

24 Stout, R. J., Salas, E., & Carson, R. (1994). Individual task proficiency and team process: What's important for team functioning. Military Psychology, 6, 177-192.

25 Driskell, J. E., & Salas, E., (1992). Can you study real teams in contrived settings? The value of small group research to understanding teams. In R. W. Swezey & E. Salas (Eds.), Teams: Their training and performance (pp. 101-124). Norwood, NJ: Ablex.

26 Artwick, B. A. (1989). Flight simulator 4.0B (Computer Program). Redmond, WA: Microsoft Corporation.

27 Prince, A., Brannick M. T., Prince, C., & Salas, E. (1992). Team process measurement and implications for training. Proceedings of the 36th Annual Meeting of the Human Factors Society, Atlanta, GA, 1351-1355.

28 Brannick, M., Prince, C., Salas, E., & Stout, R. J. (1998). Can PC-based systems enhance teamwork in the cockpit? Unpublished Manuscript.

AIAA-98-4370

AIRSIM, A DESKTOP RESEARCH FLIGHT SIMULATOR

Jaap Groeneweg, M.Sc.

National Aerospace Laboratory, NLR
P.O.Box 90502, 1006BM Amsterdam
The Netherlands
Phone: +31 20 5113595, Fax: +31 20 5113210
e-mail: groenewg@nlr.nl

ABSTRACT

AIRSIM, which is short for Avionics Integration Research SIMulator, is an integrated desktop research flight simulator, meant for avionics development and tests checkout of research simulator and research aircraft experiments, accident investigation analysis and Air Traffic Control (ATC) simulations. Developed at the National Aerospace Laboratory (NLR), AIRSIM can be used as a high fidelity, low cost flight simulator, as a computer-based trainer and as a familiarization tool for aircraft behavior and cockpit instruments. AIRSIM offers the same basic functionality as NLR's research flight simulator and has a highly configurable and flexible setup.

1. INTRODUCTION

About seven years ago it was possible to buy a voluminous graphics workstation which cost about $100,000. Nowadays it is possible to buy graphics workstations for about 10 percent of that amount with the same computing power and small enough to fit somewhere on your desktop. This development has made it possible that people can buy low-cost graphics workstations and yet run high graphical performance demanding applications while sitting behind their desk. One of these applications is AIRSIM (Figure 1).

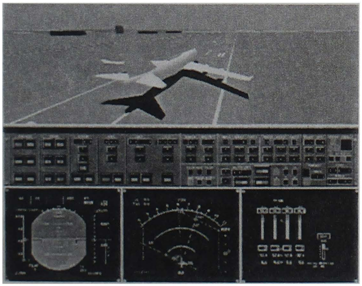


Figure 1. AIRSIM

AIRSIM is a flight simulator such as the PC-based Microsoft Flight Simulator '98 or Aerowinx Precision Simulator 1. Although built for different purposes these flight simulators have one commonality. They all depend heavily on the graphical performance of the computer.

This paper gives an overview of why NLR has developed AIRSIM, what this flight simulator is, how it works, what its characteristics and advantages are and how it can be used.

2. DEVELOPMENT PHILOSOPHY

The main reason for the development of AIRSIM is to have a low-cost research flight simulator available with the same fidelity as the Research Flight Simulator (RFS) but without the use of an entire simulator facility. Especially during the first phase of developing and testing avionics, AIRSIM is a practical tool as the developer becomes independent of the availability of the RFS.

In the second development phase porting the fully tested avionics to the RFS is easy, because common interfaces are being used. In this way, new avionics displays become available for evaluation experiments in a cost-effective way.

Day to day practice has proven that AIRSIM is not just an avionics development and testing tool. Several specific developments have made it possible that it is also used purely as a flight simulator and as an accident investigation analysis tool. What these developments are is described in the next chapter.

3. ARCHITECTURE

AIRSIM has an object oriented and distributed architecture. This means that it is built upon several modules (programs) each with a separate data structure and functions. These modules communicate via a relatively small (Ethernet) interface.
At this moment AIRSIM (Figure 2) consists of four main modules:

- Multi Cockpit Simulator (MCS)
- Electronic Flight Instruments System (EFIS)
- Generic Flight Mode Control Panel and Display Control Panel (FMPDCP)
- Out-The-Window-View (OTWV)

MCS, EFIS, FMPDCP and OTWV are also used in NLR's Research Flight Simulator (RFS) facilities.
In the near future other modules will be added, see chapter "Applications at the present and in the future".

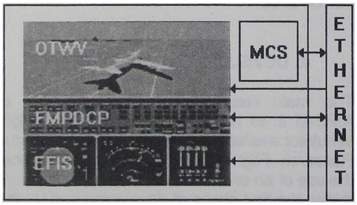


Figure 2. AIRSIM component breakdown

As shown in Figure 2 all four components communicate via Ethernet. This makes it possible to run all modules independently. You may want to run them on one workstation but it is also possible to run the modules simultaneously on several workstations under the condition that they are connected through Ethernet.

The first main module is a Multi Cockpit Simulator (MCS, Figure 3). MCS is derived from the RFS host simulation program. It is a generic flight simulator capable of simulating several kinds of aircraft. MCS as used in AIRSIM, has a Fokker-100, a Boeing 747-400, a Fairchild Metro II turboprop and an Airbus-310-like flight simulation model. Other models can be implemented easily.
For each aircraft model MCS has program routines that are capable of simulating the aircraft characteristics such as the aerodynamic model, the engine model and the gear model. Other routines simulate aircraft systems such as navigation and communication (ILS, VOR, NDB) and auto-pilot and auto-throttle systems. Last but not least MCS is equipped with a well-documented input and output interface which makes it easy to integrate MCS in AIRSIM.

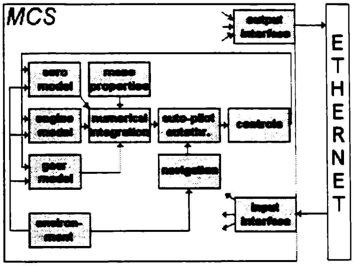


Figure 3. MCS

The second module of AIRSIM is an Electronic Flight Instruments System (EFIS, Figure 4). In AIRSIM EFIS consists of Primary Flight Display's and Navigation Display's - either from the Fokker-100 or Boeing 747-400 -, a Fokker-100 Engine Display or a Boeing 747-400 primary EICAS display. In EFIS each display is a separate module. This way it is possible to mix aircraft depended displays as shown in Figure 4 where Fokker-100 PFD and ND are combined with a Boeing 747-400 EICAS display. All displays have been built using Fokker-100/Boeing 747-400 EFIS description and Aircraft Operations Manuals (AOM). All mentioned EFIS displays are also used in the research flight simulator (RFS) and are therefore easily portable and exchangeable.

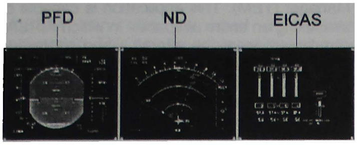


Figure 4. EFIS

The third module of AIRSIM is a generic control panel (FMPDCP, Figure 5) that interfaces via Ethernet (Figure 2) with MCS and EFIS. The FMPDCP includes a Mode Control Panel (Flight Mode Panel, FMP) to select auto pilot functions, a Display Control Panel (DCP) to control EFIS displays, a navigation source selection panel and primary and secondary controls.

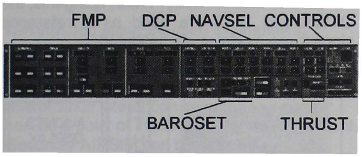


Figure 5. FMPDCP

The fourth module of AIRSIM is an Out-The-Window-View (OTWV). This OTWV is used primarily to give a view from the (Figure 7) or from a 'random' position (Figure 8). cockpit (Figure 6). In other cases OTWV can be used to give a view from a 'wingman' position (shown in Figure 1), from a 'tower' position In order to add realism OTWV is capable of simulating certain weather condition such fog and clouds. Furthermore OTWV has other visualization accessories to increase realism like mountains lakes, airfields and different graphical aircraft models. A very useful OTWV feature often used in accident analysis is a flight-path history (shown in figures 7 and 8). This is a three-dimensional presentation of the flown trajectory.

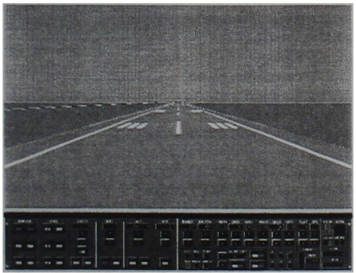


Figure 6. OTWV 'cockpit view'

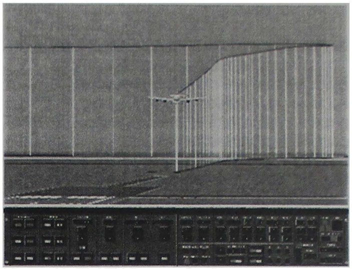


Figure 7. OTWV 'tower view'

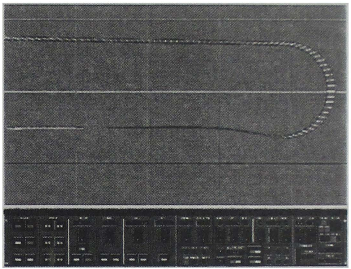


Figure 8. OTWV 'random view'

4. CHARACTERISTICS AND ADVANTAGES

AIRSIM can be characterized in several ways. Because AIRSIM has all the elements of a research flight simulator it can be characterized as a low-cost research flight simulator on your desktop. Next to that AIRSIM is characterized as a very flexible tool. This flexibility is realized in several ways.

At first AIRSIM has a modular architecture. This makes it possible to use only those modules that are necessary for your application. Additionally the modular architecture including the Ethernet communication makes it possible to run modules on different workstations and to connect new modules such as avionics displays to AIRSIM.

Secondly AIRSIM has a very flexible user interface. All modules can be controlled either using the keyboard and mouse or a special joystick or other flight controls.

Finally AIRSIM has a flexible start-up interface. This makes it possible to choose which modules will be selected and especially how MCS is used – for instance which aircraft model is used. The start-up interface can easily be adapted to your (new avionics) modules and in this way made part of AIRSIM.

5. APPLICATIONS

To a certain extent AIRSIM can be compared with MS FS '98 or PS1. All these flight simulators are cost-effective and can be used as a familiarization tool to learn aircraft behavior and operations and as a computer based trainer (CBT). But AIRSIM has more possibilities.
AIRSIM is specially made for avionics development and testing, Air Traffic Control (ATC) Simulations, testing and evaluating simulator set-ups and aircraft accident analysis. AIRSIM runs on UNIX Silicon Graphics workstations.
The next five chapters give a more detailed idea of how AIRSIM can be used.

5.1 AIRSIM IN COMBINATION WITH OTHER APPLICATIONS

To prove the flexibility and usefulness of AIRSIM, it can be connected to several other applications that are not (yet) part of AIRSIM. An important link is made with the NLR's Research Flight Management System (RFMS). This application is capable of simulating a Honeywell/Collins/Boeing like Flight Management System. Secondly AIRSIM can be connected with a Traffic and Experiment Manager (TEM). This application is capable of simulating an entire airspace - in which AIRSIM simulates one of the aircraft - and controlling the experiment. Finally AIRSIM can be interfaced with a Traffic Collision and Avoidance System (TCAS) and a Ground Proximity Warning System (GPWS) module. The TCAS module gives AIRSIM TCAS functionality and the GPWS module gives AIRSIM all the aural warnings.

5.2 AIRSIM USED AS AN AVIONICS TESTER

Most of the time AIRSIM is used for avionics development and testing. Before a new avionics display is evaluated the display is equipped with the standard AIRSIM Ethernet interface which is used by all the AIRSIM modules (Figure 9). At first the display is tested for it's basic functionality. When doing this, the OTWV and EFIS are used as a reference since these modules are validated.

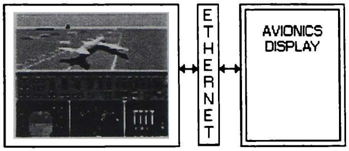


Figure 9. AIRSIM used as avionics tester

When the basic functionality test is completed, the display can be evaluated for it's operational and ergonomic capabilities.

5.3 AIRSIM USED IN ACCIDENT ANALYSIS

For accident investigation analysis AIRSIM can be used as an visualization tool in two ways: directly for technical analysis and indirectly for accident prevention.
During the technical analysis of the accident usually Flight Data Recorder(FDR) data is used to reconstruct the last few minutes of the flight. It is also possible to use MCS and reproduce the FDR data. In both cases AIRSIM provides a clear image of what has happened in the last minutes of the flight. When using MCS it may also give insight in other information such as

flight controls input.

AIRSIM can also be used in accident prevention. This can be achieved by using the visualization to compile a video animation. These animations are part of instruction video's used as illustrative medium to instruct pilots.

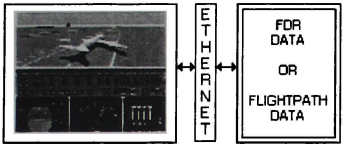


Figure 10. AIRSIM used in accident analysis

5.4 AIRSIM USED IN EXPERIMENTS

AIRSIM can easily be used in simulation experiments. This can be done in different ways. At first AIRSIM can be used as a second aircraft in flight simulation set-up. This is very useful since interaction between two aircraft can be a point of interest during an experiment. Secondly AIRSIM can be used as a 'blib - driver'. That means that - especially in ATC experiments or experiments with air traffic - AIRSIM simulates a particular aircraft. This way certain anomalies/non-normal events can be studied

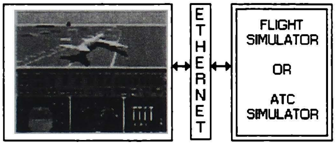


Figure 11. AIRSIM used in experiments

Just recently AIRSIM has been used twofold in a simulator experiment. Apart from the normal use during the development phase, in which important new features in the Navigation Display (ND) and Primary Flight Display (PFD) were tested, AIRSIM was also used during the evaluation trials to simulate other traffic. An important point of interest in this experiment was the interaction between two (or more) aircraft.

5.5 APPLICATIONS AT THE PRESENT AND IN THE FUTURE

AIRSIM has been and will be distributed to other NLR departments who will be using it as an avionics development and test tool and as a training tool for new employees to become more familiar with cockpit instruments and aircraft behavior. AIRSIM has been and will be used in several accident investigation analyses.
In the future AIRSIM will be fully integrated with the RFMS, the TEM, TCAS and GPWS module. After that AIRSIM will be equipped with an ARINC interface to test avionics hardware.

6. CONCLUSION

AIRSIM is very cost-effective flight simulator which can do more than other comparable flight simulators offer. The most important feature that AIRSIM offers, is the flexibility. AIRSIM can be used for much more purposes other than pure entertainment or Computer Based Training. It is mainly used for avionics development, accident analysis and experiments. Since AIRSIM is flexible in many ways it is a very user friendly flight simulator with great user potential.

AIAA-98-4371

REAL-TIME FLIGHT SIMULATION FOR GRADUATE AEROSPACE EDUCATION

M G Kellett[1], A Robinson[2], J Bauer[2] & N Baker[3]

ABSTRACT

This paper describes the concept of a low-cost piloted flight simulator for engineering officer training in the United Kingdom Royal Air Force. The Flight Engineering Systems Trainer (FEST) is a reconfigurable PC-based device capable of simulating a wide variety of subsonic fixed-wing aircraft. A 6 degree-of-motion platform supports a cockpit with dual force-feedback controls and software instrument and 3D visual displays.

The system is designed to help meet the training requirements of the Engineering Officer Training (EOT) and Advanced Systems Engineering Course (ASEC), which include components of aerodynamics, flight dynamics, stability & control and aircraft performance. The FEST system allows students to conduct simulated flight test investigations, demonstrate handling & performance characteristics, and investigate the effects of aircraft design parameters.

The FEST system was delivered to DSGT in summer 1997, and has now completed one annual cycle of students.

INTRODUCTION

The Department of Specialist Ground Training (DSGT) at RAF Cranwell is responsible for the initial training of all RAF engineering officers. Figure 1 shows the training path followed by most entrants. After completing a relevant engineering degree at a civilian university, candidates undertake Initial Officer Training (IOT) at the RAF College. Newly-commissioned officers are then located at DSGT for part 1 of Engineering Officer Training (EOT 1) which lasts 21 weeks. After a first tour on a service unit, officers return to DSGT for part 2 of EOT, which lasts another 21 weeks.

Further in-service training is supplied by specialist establishments according to discipline, and one of these is the one-year Advanced Systems Engineering Course (ASEC) at DSGT. This course is focused on providing training of selected engineering officers to develop competence in the assessment and performance of weapons and aircraft systems, and leads to Masters degree in Aerosystems Engineering which is validated by the University of Loughborough. Graduates from ASEC frequently take subsequent appointments in trials, technical intelligence, procurement and operational requirements posts.

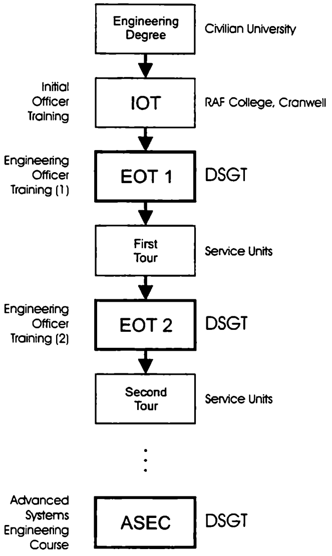


Figure 1 : RAF Engineering Officer Training

[1] Department of Aerospace, Power & Sensors, Cranfield University, Member.

[2] Deparment of Specialist Ground Training, RAFC Cranwell.

[3] IBM Global Services Ltd.

THE REQUIREMENT

The syllabus for EOT parts 1 & 2 contain significant components of aerodynamics, flight mechanics, aircraft performance, flight test and stability & control. Most of this instruction takes place in the classroom, supplemented by practicals and demonstrations in wind tunnels and laboratories. Until recently, flight demonstrations were included in the syllabus using Jet Provost, Tucano and Bulldog aircraft. This option is no longer available to DSGT, primarily due to financial restrictions.

The flight demonstrations had proved to be a most useful part of EOT however, especially considering that less than 20% of students have any hands-on flying experience, and under 4% have any formal flight training.

The current trend towards PC-based flight simulation training devices such as the Flight Navigational & Procedures Trainers (FNPTs) regulated by the JAA [1], and the Genesis system used by the Empire Test Pilot School [2], led DSGT to consider the option of procuring a simulation system to replace the flight demonstration aspect of the EOT syllabus. On further reflection, it became evident that if properly designed, such a device might be capable of contributing to the courses in ways not previously considered. For example, practical exercises could be designed around the device to introduce students to flight test procedures, and demonstrate handling qualities and dynamic characteristics of a wider variety of aircraft than real flight (including marginal and unstable configurations). In addition, more advance courses such as ASEC could perform flight trials to validate aircraft performance and compliance with operational requirements, investigate aircraft design characteristics and perform flight modelling validation and verification exercises.

The device would be required to:

- be capable of simulating a variety of aircraft configurations, including those specified by the user,
- allow hands-on use by students with flight experience, or demonstration by instructor,
- allow demonstrations in a classroom situation with a large group of students.

The primary requirement however, was that the device must be able to support the existing syllabus requirements of EOT and ASEC, since these had been formulated to meet the training requirements of the RAF.

The RAF recognises the word *simulator* as being a device for training pilots, so the name *Flight Engineering Systems Trainer* (FEST) was coined to describe the new simulation device. After a competitive tender, Data Sciences UK (now a division of IBM Global Services) was awarded a contract in 1996 to develop the system, with Cranfield University as subcontractor to supply the flight model.

THE FEST SOLUTION

The FEST system was delivered to DSGT in summer 1997, and comprises a two-seat cockpit on a 6 DOF motion platform, together with an Instructor Operating Station (IOS), computer rack and associated hydraulic and electric power services.

Figure 2 shows a schematic arrangement of the system. The two-seat side-by-side cockpit contains dual powered flying controls, instrument display on a LCD screen, and outside-world visuals on a large monitor. A pair of large video monitors mounted on stands provide a repeat of the cockpit displays for the rest of the classroom.

The IOS is the control console for the system, and has its own computer, motion control display and control panel, and a video feed from a CCD camera mounted in the cockpit. This is an important safety feature since it allows the instructor to monitor the students and abort the sortie if required. The principal components of the system are:

Computer Network

A network of standard Pentium PCs provides the backbone of the system. Apart from the IOS machine, the PCs are mounted in a rack, and separate machines are provided for motion control, visuals, cockpit (instruments and control loading) and the flight model. The machines are all linked by standard ethernet connection.

Cockpit Instrumentation

A generic instrument display is used which is modelled on the central console of a Hawk aircraft. Since the simulation is generic in nature, it was not felt necessary to allow specific modelling of individual aircraft to any accuracy. Most of the applications for FEST require little more than display of air data, attitude, heading and basic engine parameters. For more detailed off-line analysis, a whole range of flight parameters can be recorded from the IOS

Flying Controls

Dual flying controls are provided. An electric control loading system is installed on the elevator and aileron axes of the joystick. Rudder bars have a simple spring system for feel control, and the dual throttle box is a simple friction device. The dynamic control loadings are implemented using embedded microprocessor boards which have serial interfaces to the cockpit computer, which supplies force slope and trim position data calculated in the flight model.

Motion System

The FEST motion system (figure 3) is a hydraulic 6-degree-of-freedom platform of conventional configuration, developed by Intersim Ltd. A motion control computer receives displacement, velocity, acceleration and jerk information from the flight model, and applies the appropriate motion control laws to drive the six jacks. A direct control box is installed at the IOS to disengage the motion from the flight model and allow safe raising and lowering of the platform. The maximum motion system parameter are shown in table 1.

Surge	+/- 350	mm
Sway	+/- 310	mm
Heave	+/- 220	mm
Roll	+/- 30	degrees
Pitch	+/- 36	degrees
Yaw	+/- 36	degrees
Acceleration	+/- 1.0	g
Ram Velocity	+/- 0.75	m/sec

Table 1 : Motion System Parameters

Flight Model

The principal objective in designing the FEST flight model was to produce as realistic a representation of subsonic fixed-wing aircraft nonlinear dynamics as possible using a minimum parameter set. This set was constrained to use mostly the aircraft geometrical parameters (physical dimensions) and a minimum of aerodynamic data. For the most part, FEST models can be constructed using very simple estimates of aerodynamic parameters (for example, lift-curve slopes and drag coefficients) obtained from textbooks.

This approach has proved successful, and the FEST flight model has demonstrated itself capable of predicting the major dynamic characteristics of all aircraft types tested so far. Where the modelling approach is less successful is the prediction of aircraft performance, which naturally depends highly on powerplant and drag characteristics. A trimmed flight state is achieved when thrust and drag are in equilibrium, i.e. the difference between two large numbers is zero. The achieved flight condition is then numerically sensitive to errors in either side of the equation.

Flight Model Parameters. The FEST flight model is described by a set of 90 individual parameters, divided into 8 groups:

mass and inertia,
wing/fuselage geometry and aerodynamics,
tailplane geometry and aerodynamics,
fin geometry and aerodynamics,
control surfaces,
engines,
auxiliary aerodynamics (flaps, airbrakes, gear),
undercarriage.

Each group has its own page on the IOS software, and values can be entered during run-time, and even changed in flight. A blending period of 10 seconds is applied to all model changes to avoid transients.

Aerodynamic parameters consist of the physical geometry of the aerodynamic component (wing, fin etc), plus lift-curve slope and drag coefficients. These are easy to estimate from standard textbook techniques (e.g. Roskam [3], or ESDU data sheets [4]), and eliminate the need to include detailed aerodynamic data. Interaction between aerodynamic surfaces is limited to downwash effect on the tailplane. Engines are modelled as simple thrust generators (modified for altitude effects), processed through a first order lag to represent the engine dynamics.

Auxiliary aerodynamics (flaps, airbrakes etc) are modelled as incremental changes to the wing-body lift, drag and pitching moment characteristics. Deployment of auxiliary surfaces is through a rate-limited first order lag.

Diagnostic Tests. The FEST flight model is capable of correctly simulating poorly designed aircraft, or even aircraft with incorrectly entered parameters. These situations can be difficult to debug, since there are over 90 parameters to investigate.

Students are likely to find this debugging task more difficult since they are less able to correlate observed aircraft behaviour with incorrect parameters. Even for experienced investigators, the simulator cockpit is *not* the correct place to debug flight model parameters. This should be done off-line, since interpretation of aircraft behaviour is at best difficult and at worst subjective.

To assist with this task, a page has been included in the IOS software which displays a list of flight model diagnostics, which are calculated from the parameter list. The objective of the diagnostics is simply to perform a basic sanity check on the parameters. Each parameter is automatically subjected to range constraints when entered, but it is still possible to specify the wings and engines of a 747 on an aircraft weighing 1000kg. Hence the diagnostics concentrate on providing information regarding the relationships between parameters. The diagnostics included are:

- **Static margin**. This is a measure of the basic longitudinal stability of the aircraft. Errors in any one of a number of parameters (wing area, tail area, lift-curve slopes etc) will result in a silly value for the static margin.
- **Thrust/weight ratio**. Only exceptional aircraft have a T/W greater than unity, and any aircraft with T/W<0.1 is unlikely to leave the ground.
- **Wing Loading**. An aircraft with an unconventional wing loading is not likely to fly properly.
- **Cruise Speed**. This diagnostic is primarily included to estimate a sensible starting speed for the trim optimisation algorithms. It is simply the speed required to fly straight and level at a lift coefficient of 0.25, and indicates a sensible value which should lie somewhere in the middle of the expected speed range of the aircraft.
- **Max Lift to Drag ratio**. The ranges of expected L/D are well known, and outrageous values (over 100, or under 1.0) would indicate serious errors in flight model parameters.

Undercarriage parameters are particularly difficult to estimate well. FEST users are typically less interested in ground handling characteristics than the aerodynamics, but a good ground handling model is essential for satisfactory operation of the system. Particular care has been taken to provide a set of gear diagnostics to assist in parameter selection:

- **Gear natural frequency**. This measures the stiffness of the gear in relation to the mass of the aircraft.
- **Gear damping ratio**. This indicates whether sufficient damping is provided by the gear shock absorbers.
- **Gear balance.** An important parameter, this is the static load ratio from front to rear, and is the primary measure of whether the gear legs are correctly located. Too much load on the nose gear will result in an aircraft which is difficult to take off.

FEST IN USE

This section describes some of the exercises that the students undertake with the FEST system.

The EOT 2 course includes modules on aerodynamics, flight test & performance and stability & control. Students perform 6 FEST exercises in these modules, covering:

- **International Standard Atmosphere**. Students are introduced to altimeter settings, QNE, QFE and QNH. Flight demonstrations during changes of barometric pressure are included, and students perform altimeter calcuations.
- **Airspeed**. Relationship between IAS, EAS and TAS is demonstrated by flying at fixed speed and heading over a measured distance.
- **Climb Performance**. Concepts of minimum drag speed and maximum excess power speed from lectures are reinforced by practical flight tests to measure aircraft climb performance.
- **Dynamic Modes**. Students make theoretical predictions of the five basic aircraft modes (Phugoid, short period, dutch roll, roll siubsidence and spiral modes), and are then required to measure them in flight. This gives a good opportunity for students to gain hands-on experience of the effect on handling qualities of these modes, and also experience the modes of aircraft which they would not normally have access to.
- **Neutral Point Measurement**. Having covered the theory of longitdunal stability, a practical exercise is performed to measure the stick-fixed neutral point. Having determined this (by extrapolating a graph of Stick/Cl slope vs c.g. position), students are then invited to place the c.g. at the neutral point and experience the resulting flying characteristics!
- **Level Flight, Stall & Stall Speed**. Students demonstrate to themselves the change in pitch attitude (Cl) required to maintain level flight at different speeds. Stalls, and recovery procedure, are then demonstrated by cutting power and maintaining altitude. An investigation of stall speed then follows by varying flap settings, aircraft mass andwing are, and comparisons to predicted values are made. Finally stall in accelerated flight (2g turn) is demonstrated and compared to level flight.

For Advanced Systems Engineering Course (ASEC), the students undertake a group performance assessment exercise. This involves a series of flight tests to establish the aircraft flight envelope, take-off, climb and landing performance. These are then compared to the Operating Data Manual figures for an in-service aircraft of a similar type. The final part of the exercise is an assessment of the handling qualities characteristics.

ASEC students are also required to perform an individual project, which is a major contribution to their overall assessment. The FEST facility offers opportunitiesfor project investigations that were not previously possible. The first of these has now been completed, and was an assessment of the ability of the system to simulate the characteristics of a Grob 115A light aircraft [5].

Conclusions

A real-time piloted flight simulation facility has been developed to support the training of RAF engineering officers at the Department of Specialist Ground Training. The system is used for a variety of roles, from preformance measurement through handling qualities to demonstration of flight test procedures. For advanced courses, the system acts as a virtual laboratory aircraft for students needing considerable quantities of flight test data.

The system has been in service for over 12 months and has proved to be entirely satisfactory in replacing the flight demonstrations previously in the syllabus. In many cases, the simulator provides additional functionality since it has greater accessibility to the students, and allows exercises to be inserted at pertinent points during the training programme.

References

1. *Flight & Navigation Procedures Trainers*, JAR-STD 3A, Joint Aviation Authority, 1996.

2. *Genesis 2000 Engineering Flight Simulation System*, VEDA International Corporation.

3. Roskam, J. *Airplane Design*, Roskam Aviation & Engineering Corporation, Ottowa KA, 1997.

4. *ESDU Engineering Data*, Engineering Sciences Data Unit, 251-259 Regent St, London, W1R 7AD, UK.

5. Sansom, A M. "*Reality or Fantasy? An Investigation into the Capabilities of the Flight Engineering Systems Trainer*", MSc Thesis, Loughborough University & RAF College Cranwell, 1998.

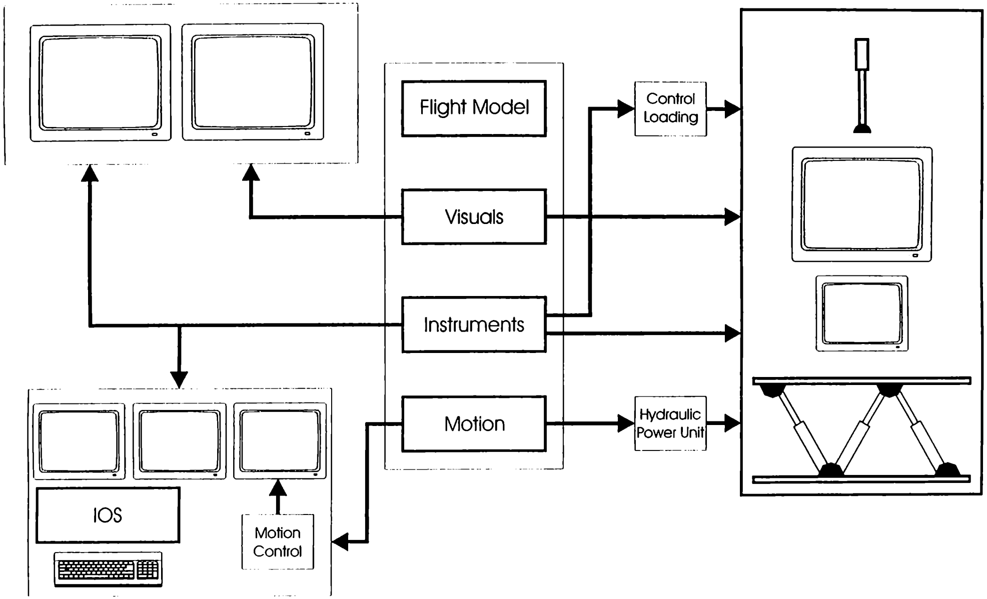


Figure 2 : FEST Hardware Schematic Arrangement

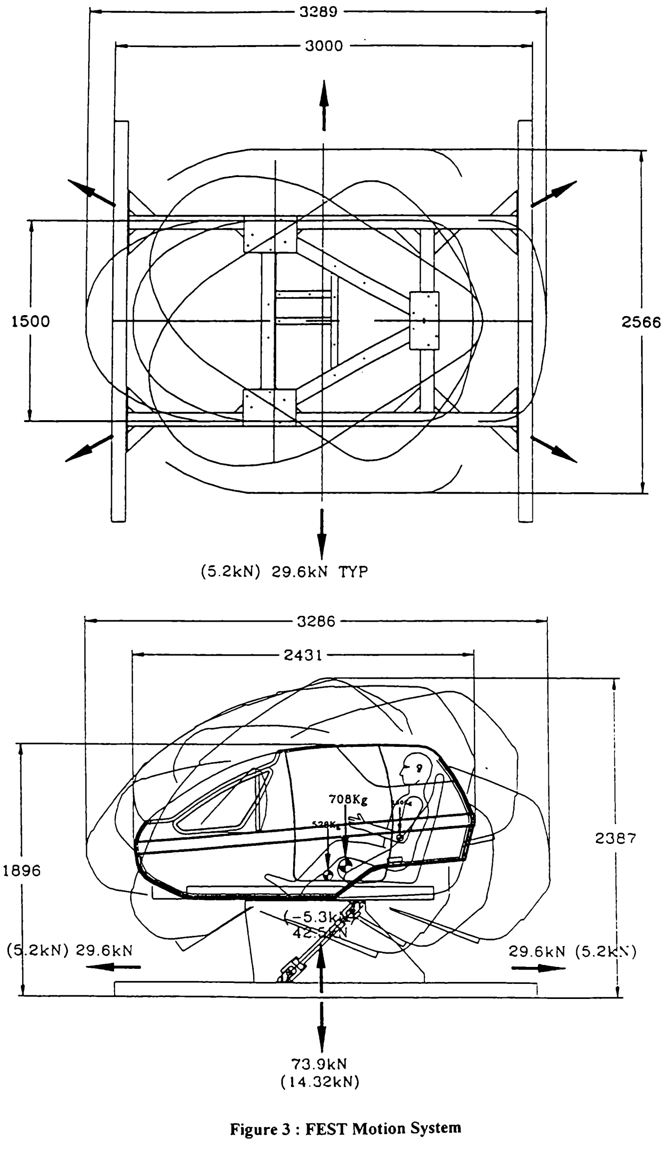


Figure 3 : FEST Motion System

AIAA-98-4372

A PC-BASED SYSTEM FOR VIRTUAL NAVIGATION OF A WING-IN-GROUND EFFECT CRAFT

K.V. Rozhdestvensky and M.A. Mikhailov
State Marine Technical University
Saint-Petersburg, RUSSIA

ABSTRACT

The paper is focused on a description of PC based system supporting virtual navigation of a wing-in-ground (WIG) effect craft, one of the fast sea transportation alternatives of the next milenium.

The system incorporates several modules, some of which are still under development. The module of dynamics of the vehicle features six-degree of freedom equations of motion with account of effects of wind-wave perturbations and action of controls. The module of virtual navigation provides 3D representation of the vehicle, other moving and still objects (conventional and fast ships, islands, buoys, beacons etc.) in a realistic marine environment.

The system incorporates effects of deterioration of visibility due to fog or darkness. It also enables to imitate lights carried by participants of the traffic, and sounds produced by their syrenes and machinery. When simulating navigation of large a nd very fast vehicles it is indispensable to model performance of radars securing more time for decision making for collision avoidance. Together with a 3D visual traffic scene there is available a "radar window". The system enables a user both to choose a WIG vehicle and create a navigational and sea environment prescribing courses and speeds of participants of traffic.

Authors' Biographies

Kirill V. Rozhdestvensky - Professor, D.Sc., CEng, FIMarE. Graduated from Saint Petersburg Marine Technical University (MarTech), in 1969 as an engineer in hydroaerodynamics. He obtained Ph D and D Sc degrees from MarTech in 1972 and 1982 on Advanced Marine Vehicles. His present position is the Chairman of the Department of Applied Mathematics and Mathematical Modeling (DAMMM) and Dean of the Faculty of Shipbuilding and Ocean Engineering of MarTech. From 1984 through 1991 – member of High-Speed Marine Vehicles Committee of the International Towing Tank Conference. From 1994 till present time – Deputy Chairman of the International Coordinating Committee on Ekranoplans. In March 1998 he has been awarded Denny Gold Medal of the Institute of Marine Engineers (London) for his paper "Ekranoplans-The GEMs of Fast Water Transport".

Mikhail A. Mikhailov graduated from Saint-Petersburg North-West Polytechnic Institute as an engineer in electronics in 1990. He has experience and inventions in the field of signal measuring and processing instrumentation. From 1990 his engineering and research activities have been mostly related to multimedia and virtual reality/virtual environment technologies.

He participated in several educational projects connected with development of interactive learning software packages with use of new information technologies. At present he is the Head of the Laboratory of Virtual Modeling of DAMMM of MarTech.

1. Introduction

Though the first ground effect machine was designed and built by a Finnish engineer Kaario back in 1935, the scale of the technology was demonstrated in late sixties when the 500 ton

KM, better known in the West as the "Caspian Sea Monster", flew at a speed of 500 km/h a meter or so above the water. Further developments showed certain perspectives of utilization of wing-in-ground effect craft both for naval and commercial missions[1,2].

Essentually, the ground effect displays itself in increase of lift-to-drag ratio when a lifting surface moves in proximity of an interface. Technical difficulties in implementing the technology are connected with: large power needed for take off, specifics of requirements for static stability of longitudinal motion in presence of an underlying surface, *etc.*

The speed of WIGs by far exceeding that of conventional ships and even that of hovercraft and hydrofoil craft, safe navigation of these vehicles in existing sea traffic turns out quite a nontrivial task[3,4]. As a matter of fact, the goal of the system is to virtualize navigation with aviation speeds at the sea level. This task can be solved by means of synthesis of appropriate mathematical models with up-to-date methods of virtual reality and 3-D graphics[5-6].

2. General Description of the System

The system for assessment of navigational safety of a wing-in-ground effect vehicle incorporates four principal parts:

- Module of Dynamics of the Craft
- Editor of Parameters of a Vehicle
- Editor of Sea and Navigational Environment
- Editor of Testing Scenario and the Tester Itself

2.1. Peculiar Features of Dynamics of Motion of a WIG Vehicle

The module of dynamics of the vehicle (still under development) makes use of the linearized equations of motions of a wing-in-ground effect. Detailed description of dynamics of motions of WIGs can be found in literature[7,8]. Here ,due to the lack of space, only some pecularities of dynamics and one concrete example of relevant equations will be discussed. It should be noted that in general case of medium ground clearances 3-D dynamics of WIG craft is characterized by a much stronger coupling of parameters of longitudinal and lateral motion than for an airplane (in unbounded flow case). However, for small relative clearances (extreme ground effect), when the efficiency of WIG vehicles is most pronounced, the coupling of longitudinal and lateral motions can be neglected. In the concrete case of perturbed longitudinal motion the linearized equations of dynamics of an uncontrolled WIG vehicle can be written as[2]

$$\mu\frac{dU'}{dt} = (C_t^U - 2C_{D_o})U' -$$

$$C_D^\theta\tilde{\theta} - C_D^h\tilde{h} - C_D^\theta\dot{\tilde{\theta}} - C_D^{\dot{h}}\dot{\tilde{h}} \tag{1}$$

$$\mu\frac{d^2\tilde{h}}{dt^2} = 2U'C_{y_c} + C_y^\theta\theta +$$

$$C_y^{\dot{\theta}}\dot{\tilde{\theta}} + C_y^h\tilde{h} + C_y^{\dot{h}}\dot{h} \tag{2}$$

$$\mu i_z\frac{d^2\tilde{\theta}}{dt^2} = y_t(2C_{D_o} - C_t^U)U' +$$

$$m_z^\theta\tilde{\theta} + m_z^h\tilde{h} + m_z^{\dot{h}}\dot{\tilde{h}} + m_z^{\dot{\theta}}\dot{\tilde{\theta}} \tag{3}$$

In the above equations U' represents a relative perturbation of cruise speed, C_t^U is a derivative of the thrust coefficient with respect to cruise speed, C_{D_o} is a static drag coefficient, y_p is vertical distance of the thrust line from the vehicle's center of gravity. As earlier, all quantities are rendered nondimensinal with use of cruise speed U_c and root chord C_o, μ is relative density of the vehicle defined by the formula

$$\mu = \frac{2M}{\rho S C_o} \tag{4}$$

where $\tilde{h}, \tilde{\theta}$ represent perturbed relative ground clearance and pitch angle, M is the vehicle's mass in cruise, S is a wing's reference area, ρ is specific density of air and i_z is a coefficient of the longitudinal moment of inertia of mass, determined by means of a relationship

$$i_z = \frac{I_z}{MC_o^2} \tag{5}$$

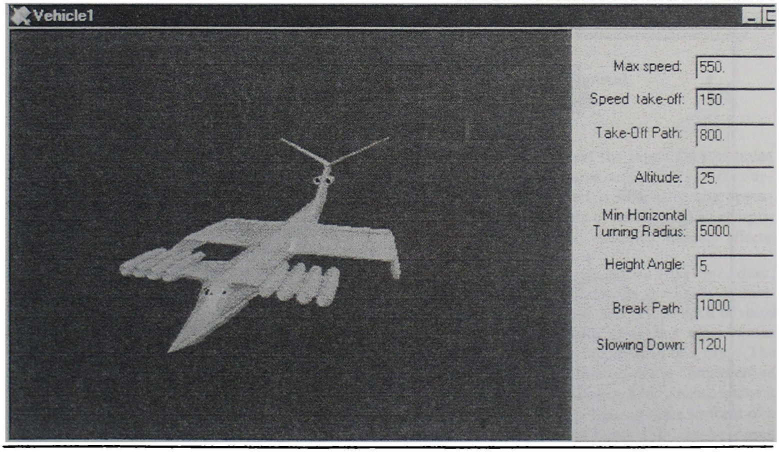
where I_z is the longitudinal moment of inertia of mass with respect to the center of gravity.

It is worth noting that wing-in-ground effect vehicles have quite particular properties of static stability of longitudinal motion. A distinction from the airplane is that unless aerodynamic configuration of the vehicle is designed properly, it is impossible to secure stability by appropriate choice of position of the center of gravity. Instead of one aerodynamic center, namely, that of the angle of attack, which is characteristic of airplanes, the WIG's stability depends on reciprocal position of two aerodynamic centers (that of pitch and that of height). According to Irodov's[7] criterion to secure static stability of longitudinal motion one has to configure the wing-in-ground effect vehicle aerodynamically in such a way that the center of height should be located upstream of the center of pitch. This can be achieved in different ways, for example, by using a large highly mounted tailplane, operating out of the ground effect zone, or by using a tandem scheme with appropriately selected parameters of fore and aft wing elements, or by means of special design of foil sections of the main wing.

2.2. Editor of Parameters of a Vehicle

This module enables either to select a vehicle from existing ones or to create a vehicle

Selecting (creating) a vehicle signifies adoption of its geometry and corresponding realistic appearance and assigning a set of associated properties. 3-D models of the craft are generated prior to execution of "tests". The vehicle's associated properties may include aerodynamic coefficients, entering the equations of motion, relevant characteristic sounds (sirens, engine noise) and lights. For assessment of navigational safety of a wing-in-ground effect vehicle it is often sufficient to introduce descriptive properties of steerability and manoeuvrability, such a a minimal radius of circulation, height ceiling, speed of climbing when performing "dynamic jump" , stopping distance, cruise speed and ground clearance. The virtual mock-up of the vehicle and its attributes can be stored on the hard disc for subsequent use. Figure 1 illustrates the procedure of selection of a vehicle (in this case the *Caspian Sea Monster*) and editing of its parameters.



Max speed: 550.

Speed take-off: 150

Take-Off Path: 800.

Altitude: 25.

Min Horizontal Turning Radius: 5000.

Height Angle: 5.

Break Path: 1000.

Slowing Down: 120.

Fig. 1 Illustration of the procedure of selection of a vehicle and editing of its parameters.

2.3. Editor of Sea and Navigational Environment

This module of the software is intended for generation of the basin for testing of the previously stored mock-up of the vehicle. This editor employs two formats of representating of information: the map of the basin and a 3-D window. Editing is performed with on the "map screen". Control of the view of the sea environment under editing is carried out on the 3-D window. In the current version of the system the map can have maximum dimensions of 300 km by 300 km, so that the "trials" can be conducted of both small and large wing-in-ground effect vehicles. Dimensions are prescribed when starting a new map. Simultaneously, general type of the basin (deep or shallow) is introduced. Further on, using the instruments of the "tool bar", one can designate location and configuration of the regions of deep and shallow water, create islands, risk areas (rocks, sunekn ships, *etc.*). The landscape of the coastline can be constructed, including (albeit currently polygonal) hills, as well as buildings and structures, selected from previously stored libraries of 3-D objects. The later can contain ensembles of objects, related to a ceratain city or port (e.g. New York) and can be added when necessary. The libraries also include 3-D models of beacons, navigational buoys and other objects. They have a format of standard DLL resource file and can be developed by users with help of C++ programming language. Displacing and turning the camera icon on the map, one can control the appearance of the environment under construction in the 3-D window. Further on the created sea environment can be "stored" on the hard dsc for subsequent use. Figure 2 demonstrates editing of navigational environment, namely "building" of islands and coastline landscape as well as placing of buoys and beacons. Figure 3 provides a camera view on a local scene within the created environment.

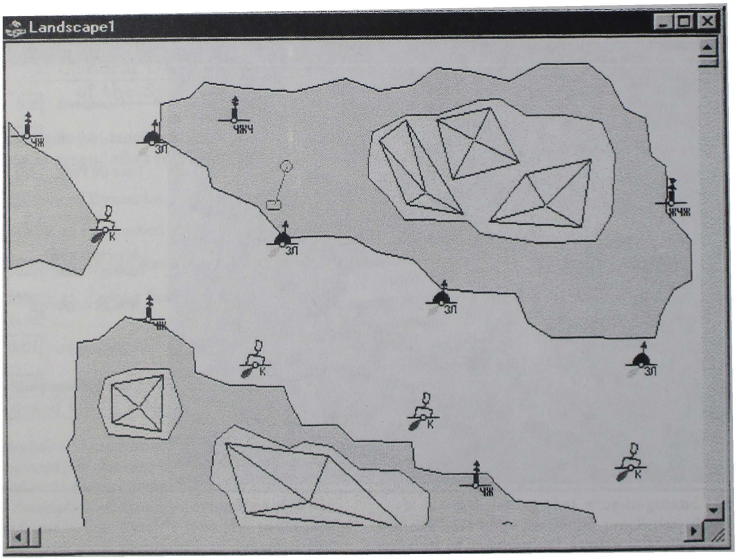


Fig. 2 Editing of navigational environment.

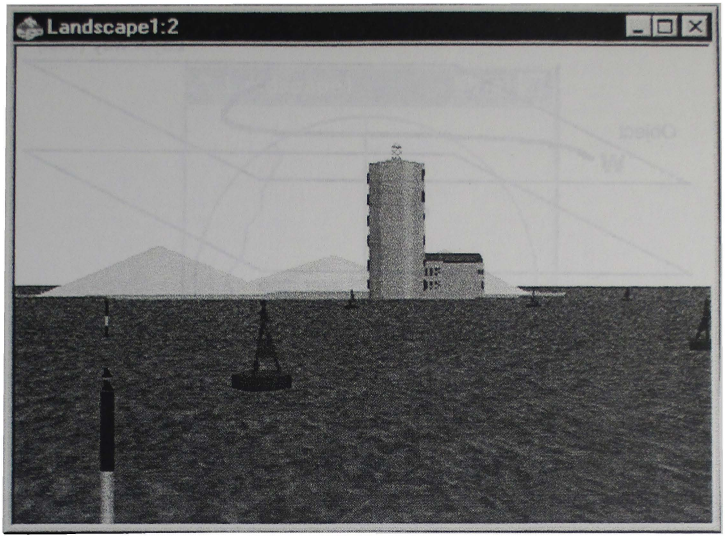


Fig. 3 A camera view on a local scene of the created environment.

2.4. Editor of Testing Scenario and the "Test Player"

This part of the software uses as an input the files generated by the two editors described above. In a "map layer" there is loaded a sea basin, developed previously. In the "trajectory layer", created by means of the editor of testing scenario, one loads a mock-up of the vehicle (active object) to undergo the test. Figure 4 illustrates a "map layer" and a "trajectory layer" of the editor of testing scenario. Beside the active object there can be introduced other inhabitants of the scene together with their associated properties. These can be displacement and aircushion ships, other wing-in-ground effect craft. Parameters of motion of these other objects are prescribed by means of tracing their routes on the maps with prescription of points of turning, course angles and cruise speeds on each part of the route. For a vehicle undergoing test the input data includes initial coordinates, course angle and speed. Further on the active object is steered by the operator. A set of parameters of testing scene incorporates factors related to visibility, namely effects of the fog and light conditions (daytime, dusk, darkness). In dusk conditions the lights of navigational buoys, beacons and cruise ligth of ships are enabled. The editor of testing scene works with the data represented in form of a map. The second input window resembles the "Explorer" of Windows 95. This latter window incorporates a hierarchie of (variable) parameters in form of a tree. The testing scene input data can be stored on a hard disc and executed when necessary. In order to activate the preliminarily "constructed" testing scenario the use is made of separate part of the software which can be dubbed "a test player". In the test execution mode representation of current information can involve up to four windows

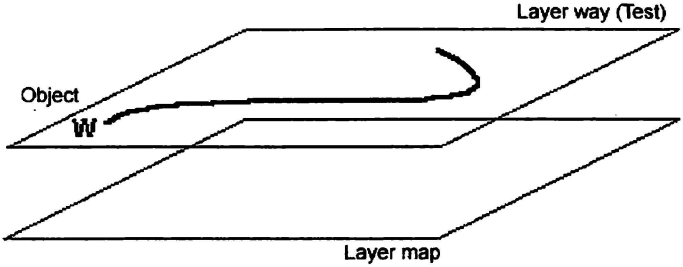


Fig. 4 A "map layer" and a "trajectory layer" of the editor of testing scenario.

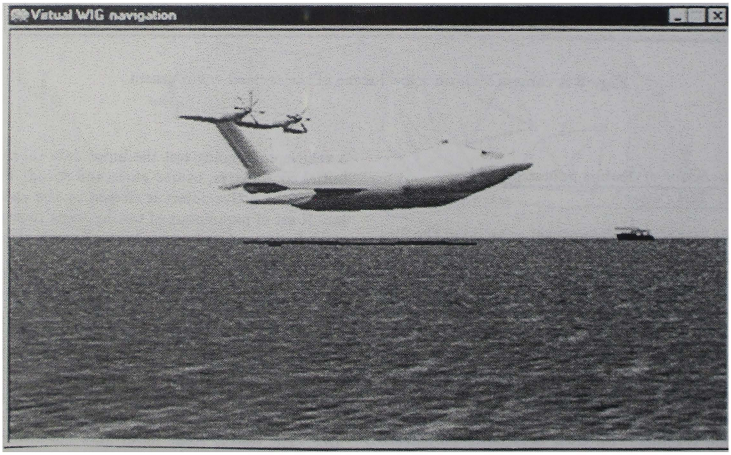


Fig. 5 A view of a testing situation for a large wing-in-ground effect vehicle.

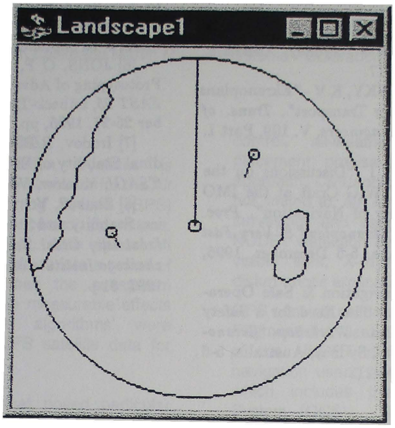


Fig. 6 The radar window.

simultaneously. it is evident that the more windows are activated at the same time, the slower flows the conditional time of the test. The aforementioned windows can reflect the following information: 3-D view of the vehicle undergoing the test, the map of the basin similar to the one discussed above, radar chart and a set of current test parameters in digital form. Figure 5 presents a view of a testing situation for a large wing-in-ground effect vehicle (in this case a 250-passenger MPE-200 of Russian design). The 3-D view of the vehicle serves to carry out visual control of the flight during the test. This control is almost always needed, because (in this version) handling of the vehicle is performed by the operator manually with help of a joystick and / or PC keyboard. The "map window" shows the region of the test and helps to evaluate current situation. Location of the vehicle under testing is always in the center of the window with standard orientation of the map (north on the top). The map is scrolled as the vehicle advances, showing current navigational situation. The radar window imitates representation of the testing scene on a radar screen (see Figure 6) with the vehicle under testing located in its center with the course lines extending upwards. The scale of the radar window can be selected based on preliminarily prescribed data. The "parameter window" shows current magnitudes of time dependent parameters, so that the user is able to check the speed, ground clearance, heel angle, distances form other selected objects, *etc.*

The software is written in Visual C++ 4.0 with use of Direct X and is intended for Windows 95 operational system and on PC platforms of Pentium 166+ with 32 Mb+ of RAM and a graphics accelerator of Diamond FIRE GL 1000 PRO. The sound effects are reproduced with use of the Sound Blaster 16 or AWE-32 card types.

4. References

[1] KIRILLOVIKH, V N -"Russian Ekranoplans", *Proc. of the Int. Workshop on "Twenty-First Century Flying Ships* , Sydney, Australia, 7-9 Nov., 1995, pp. 71-117.

[2] ROZHDESTVENSKY, K V - "Ekranoplans-The GEMs of Fast Water Transport", *Trans. of the Institute of Marine Engineers*, V. 109, Part 1, 1997, pp 47-74.

[3] BOGDANOV, A I -"Discussions on the Operational Aspects of WIG Craft at the IMO SubCommittee on Safety of Navigation", *Proc. of the Int. Workshop "Ekranoplans & Very Fast Craft"*, Sydney, Australia, 5-6 December, 1996, pp.213-229.

[4] MAYER, L -"Navigation & Safe Operation of Very Fast Craft. The Need for a Safety Case?", *Proc. of the Int. Workshop "Ekranoplans & Very Fast Craft"*, Sydney, Australia, 5-6 December, 1996, pp. 194-212.

[5] MILLER, E R, FITCH, M -"The potential Application of Virtual Reality Based Simulators to Ship Handling and Marine Operations", *Proc. Int. Conference MARSIM'96*, Copenhagen, Denmark, 9-13 September, 1996.

[6] JONS, O P, SCHAFFER, R L -"Virtual Prototyping of Advanced Marine Vehicles", *Proc. FAST 95*, Lübeck-Travemünde, Germany, September 25-27, 1995, pp. 563-573.

[7] Irodov, R.D. (1970) "Criteria of Longitudinal Stability of Ekranoplan", *Ucheniye Zapiski TSAGI*, Moscow, Vol. 1, No. 4, pp. 63 - 74.

[8] Zhukov, V.I. "Pecularities of Aerodynamics, Stability and Steerability of an Ekranoplan", *Izdatelsliy Otdel Tsentralnogo Aerogidrodinamicheskogo Instituta im. N.E. Zhukovskogo*, Moscow, 1997, 81 p.

AIAA-98-4375

HIGH FIDELITY GPS SATELLITE SIMULATION RESULTS

Mark Baker, Lockheed Martin Mission Systems – Colorado Springs, CO

Abstract

Lockheed Martin Mission Systems is developing a high fidelity simulator for the Air Force's Global Positioning System (GPS) satellite command and control operations. To accomplish this goal, each spacecraft subsystem was broken down into a series of algorithms that describes the subsystem dynamic behavior. The measurable effects produced by these algorithms were compared to actual GPS satellite data for algorithm validation.

Several subsystems that posed particular difficulty include orbit determination, thermal control, and reaction control. The primary algorithms used to simulate these subsystems are included herein as well as a discussion pertaining to issues surrounding the development of these algorithms. Finally, some of the required anomalies are discussed along with their relation to the derived algorithms.

Introduction

GPS is a spaced-based radio navigation, time distribution and nuclear detonation (NUDET) detection system. Its mission is to provide precise, continuous, all-weather, three-dimensional (3-D) position, velocity, timing, and NUDET information to properly equipped air, land, sea, and space-based users. The system serves various military and civil operations as a highly accurate positioning and navigation data reference. Military operations include, but are not limited to, enroute navigation, low-level navigation, target acquisition, close air support, missile guidance, command and control, all-weather air drop, sensor placement, precise survey, and instrument approach. Civil applications include, but are not limited to, air, land, and sea navigation, surveying, and search and rescue. As a NUDET detection system (NDS), it detects visible light, X-rays and electromagnetic disturbances emanating from NUDETs.

GPS consists of three segments: the Space Segment; the User Segment which consists of the NUDET detection users and the navigation users; and the Control Segment which includes the Operational Control System (OCS). The OCS includes the Master Control System (MCS), multiple Ground Antennas (GAs), and multiple Monitor Stations (MSs).

The GPS OCS utilizes GAs and MSs to monitor and control the GPS constellation and to receive GPS Navigation data. Ground Antennas contain equipment to receive a single GPS S-Band State-of-Health Telemetry downlink and to uplink satellite commands and Navigation Uploads to a GPS satellite. Each Monitor Station contains a GPS L-Band Antenna and a twelve channel receiver which allows it to process the Navigation Telemetry downlink from multiple GPS satellites simultaneously.[1]

The Need for a GPS Simulator

The Air Force operations squadron that controls the GPS satellites from the MCS holds regularly scheduled training sessions for crews that perform the actual operations. Until quite recently, most of these training sessions have included a large degree of 'paper training', as they had no way to adequately simulate effects of commands that get issued to the satellite vehicles, nor did they have any way to introduce anomalous conditions into the training system. The GPS High Fidelity Simulator (herein referred to as the Simulator) will greatly enhance their ability to properly train crew members in day-to-day functionality and in handling anomalies that occur. It will also aid in the resolution of problems for which there is no established operating procedure. For instance, if a problem occurs on a vehicle, there may be a variety of proposed solutions for fixing it, but the effectiveness and the secondary effects of a proposed solution may be unknown. High fidelity simulations will produce data to aid in selecting the best proposal. In addition, the simulator may be used for testing new MCS software releases and maintenance fixes.[1]

Satellite Subsystem Algorithm Development

The algorithms developed for the simulator have varying levels of fidelity. Because of this, different algorithms posed greater difficulty for simulation. In several instances, partial models were derived only to find out that one small piece of data or a constant needed to complete the algorithm was either unavailable or the manufacturer was unwilling to divulge it. For these cases, reverse engineering was performed to bridge the gap between theory and actual subsystem performance as measured on orbit.

The figures depicting the results of these algorithms are included at the end of this paper. These plots were generated from data output from the simulator software that implements these algorithms. It should be noted that the constants and variables used in these equations are also listed and defined at end of this paper.

Orbit

The orbit prediction problem has been solved many times in the past several decades. The simulator team chose to use established methods for orbit propagation. The models used include the Earth's gravity, solar gravity, lunar gravity, and solar pressure. The effects of solar and lunar gravity are computed as point mass effects. The effects of the Earth's gravity is computed based on the WGS84 model and uses zonal, tesseral, and sectorla harmonics with an eighth degree and order. The effects of thruster firings are also included in the orbit propagation and will be discussed in greater detail. Figure 1 shows the simulator orbit prediction as compared to ephemeris data generated by the National Imagery and Mapping Agency (NIMA). Since the NIMA data has been "smoothed" based on several ranging measurements to a GPS satellite, differences are expected. A new orbit prediction is computed every 1.5 seconds. This propagation period was chosen to coincide with the period needed for L-Band ranging calculations performed at an MS. With this short period, a quick yet accurate integration method was chosen. The two methods used in the simulator are the eighth order Runge-Kutta as a starter to a Predictor-Corrector method. With this combination, the accuracy requirements in addition to the performance constraints have been met. Performance monitors were put in place and measured an average propagation cycle of 12ms of CPU time. While this time is the largest of all the simulated subsystems, it clearly does not consume a significant amount of computer resources.

Thermal Control Subsystem

The Thermal Control Subsystem (TCS) computes the temperatures of various spacecraft surfaces and components. It also simulates the heaters needed to maintain a proper thermal range for various components. TCS is modeled using several different techniques.

The first is a high fidelity energy balance for the spacecraft external surfaces and solar arrays. The governing differential equation used to compute the temperature of a given surface is given by:

$$\frac{dT_{SV}}{dt} = \frac{1}{mc}[\dot{Q}_{Joule-heating} + \alpha_{SV} A_{\perp} KI_{Sun} - \sigma A_{SV} \varepsilon_{SV} T_{SV}^4 + \\ \sigma A_{SV}\left(F_{SV-Space}\varepsilon_{Space}T_{Space}^4 + F_{SV-Earth}\varepsilon_{Earth}T_{Earth}^4\right)]$$

For the case of solar arrays, adjustments are made to the equation to compensate for the radiation of energy from two surfaces and the absorption of solar energy from one. A separate differential equation is evaluated for each surface of the spacecraft. These equations are then numerically integrated to produce a new temperature state for each surface. Figure 2 shows the integrated surface temperatures over a 10 hour period. The semi-synchronous orbit of the GPS satellites can easily be seen in the thermal data. Notice how the +X surface oscillates in temperature with a period of approximately 12 hours.

The second method for thermal simulation is used to compute the temperatures of the components on the interior of the spacecraft. For GPS spacecraft, many components are mounted on the internal surface of the panel that makes up the exterior structure of the spacecraft. When a particular exterior surface is illuminated, heat is transferred to the components mounted to the other side of this surface by conduction. Therefore, a simple thermal model can be constructed using the cosine of the angle of solar radiation (0° is defined as the surface normal) as a scalar to a predetermined temperature range. Separate ranges are given for operational states and non-operational states of the component. The eclipse factor, *K*, is also included to account for any decrease in solar radiation due to an eclipse. The value of *K* ranges from 0, indicating full eclipse, to 1, indicating full sun. The equation for the modeling of internal temperatures is:

$$T = K(T_{hi} - T_{lo}) \cos(\theta) + T_{lo}$$

A full scale thermal model that divides the spacecraft into nodes and solves the heat equation at each node was considered. This method was not incorporated due to the number of nodes needed to encompass every spacecraft component. The solution to this system is computationally intensive and is not well suited for a real-time simulation.

One of the component heaters that is modeled is the battery heater. This heater is controlled by a thermostat with a high and low set-point. The equation for the battery temperature is given by:

$$\frac{dT_{battery-N}}{dt} = \frac{1}{m\,C}(\eta_{heater} Q_{heater} + Q_{heat-of-reaction} - A_{battery} k_{SV} \frac{T_{battery} - T_{SV}}{L})$$

Here, it is interesting to note the dependence of the battery temperature on the temperature of the mounting surface, T_{sv}. This interaction leads to complex thermal dynamics. A plot of battery temperature and space vehicle mounting surface temperature versus time is shown in Figure 3. The sudden drop in battery temperature is due to the battery heaters turning off. The upper set point for the battery heaters is +8°C. Although the battery heaters turn off at the desired temperature, the battery mounting surfaces continue to point to the sun and continue to heat up. Since the heat flow due to conduction is still greater than the heat flow due to radiation, the batteries continue to heat up.

The fourth order Runge-Kutta integration method was chosen for the thermal subsystem since the equations are quite stable and precision is not of utmost concern. Also, the period for these calculations was chosen to be 60 seconds due to the slow change in the thermal state of the spacecraft.

One anomaly required by the user is to fail a battery heater on or off. When the "Battery Heater Failed On" anomaly is injected, the battery heater will fail to turn off when the battery temperature reaches the high set-point for the battery thermostat. This anomaly can be used to observe the battery current output and its effect on the spacecraft power loads. When the battery heater is failed off, the battery heater fails to turn on when the battery reaches the low set-point. This anomaly is also used to observe the results of spacecraft electrical performance when the battery temperatures fall below the design limits.

Reaction Control System

The simulation of the Reaction Control System (RCS) includes the spacecraft thrusters as well as the propellant tanks, and lines. To ensure that the thruster efficiency is maximum, the propellant is heated using a catalyst. The catalyst is heated using a heater that is nominally switched on prior to thruster firing. The model for the catalyst heaters is contained in TCS.

For the thruster model, the standard rocket equation is used.[1]

$$T = \dot{m} V_e + (p_e - p_a) A_e$$

Knowing the nozzle exit area ambient pressure, and estimating the nozzle exit pressure, the solution of the thrust equation only involves the calculation of the mass flow rate and the exit velocity. Through several simplifying assumptions, such as one-dimensional, adiabatic flow, and a calorically perfect gas, an equation to model the exit velocity can be obtained.[2]

$$V_e^2 = k R_{gas} T_c \frac{1 - \left(\frac{p_e}{p_c} \right)^{\left(\frac{k-1}{k} \right)}}{k - 1}$$

Making similar assumptions, an equation for the mass flow rate can be derived.[1]

$$\dot{m} = p_c A_t \left[\frac{k}{R T_c} \left(\frac{2}{k + 1} \right)^{\frac{k+1}{k-1}} \right]^{\frac{1}{2}}$$

This still leaves the problem of computing values for the pressure and temperature of the combustion chamber. The temperature of the combustion chamber can be difficult to obtain since it varies with the catalyst heater performance and the friction flow of the propellant during a thruster firing. From on-orbit data, it can be seen that the thruster chamber temperature heats up to its nominal temperature in less than one hour using only the catalyst bed heaters. Once frictional heating is employed during a thruster firing, the chamber heats up to the nominal temperature in approximately 10 to 20 seconds. The following set of equations has been derived to model the thermal behavior of the GPS satellite thrusters.

$$h = p \left(\frac{\dot{m} V_c^2}{2} \right) + q$$

$$\dot{T}_c = \frac{1}{m\,c} \left[\dot{Q}_{CatbedHeater} + \frac{h V_c^2 \dot{m}}{2} + hA(T_H - T_c) \right]$$

Figure 4 shows the rise time for the chamber temperature for the nominal scenario where only the catalyst bed heater is used. Figure 5 then shows the rise time when the thruster firing frictional flow is used to heat the chamber. As can be seen in both figures, the simulated thrusters behave very closely to the actual GPS satellite thrusters.

Next, the chamber pressure must be computed. Having an accurate model for the chamber temperature and knowing the relationship between the propellant feed pressure and the thrust provided, an equation can be derived to compute the chamber pressure. This equation is given by:

$$p_c = a E_{T_c} P_{\Pr op}^b$$

Putting these equations together to form the reaction control model, one can produce a plot of thrust vs. propellant feed pressure. The spacecraft controllers use a chart of this type created from on-orbit data for orbit maneuver planning purposes. Since the propellant tank pressure is monitored in telemetry, looking up the corresponding thrust will give the satellite engineer the proper thrust performance to expect during a thruster firing. The comparison between the on-orbit data and the equations used in the simulator are shown on the plot in figure 6.

The execution period of this subsystem is dependent on the thruster firing state. If there is no thruster firing in effect, the period is 60 seconds. During this time, propellant tank pressures, which are not included in this discussion, are computed based on changes in tank temperature. If a firing is in effect, the model execution period can be as short as 80ms. This period is required to keep the attitude control system stable. At this time, no performance data is available for this model.

Several anomalies have been required for the reaction control model. A simpler set of equations was considered for the simulation of the reaction control system but these methods did not allow for the interaction of some of the more complex anomalous conditions.

One simulated condition is the propellant tank leak anomaly. For this, the user specifies the time it takes to bleed the tank from the current pressure to zero. An exponential equation for the mass flow rate is then solved based on this time.

$$\dot{m}_{leak} = 0.01\left(1 - e^{\frac{m_H}{200T_{empty}}}\right)$$

Figure 7 shows the results of this equation for a T_{empty} of 7 days. With this method, the operations training personnel can simulate for training purposes, a realistic propellant leak scenario.

Another scenario that can be trained is the "Thruster Failed to Turn Off" anomaly. For this case a thruster will remain on even though it has been commanded to shut off. This results in more thrust than was expected causing changes to both the orbit ephemeris and the spacecraft attitude. The spacecraft operator must recognize this failure and isolate the thruster from the propellant tank before the satellite state has been altered such that it cannot recover.

Real-Time Performance

Great care has been taken to ensure that the entire GPS satellite simulation will execute with real-time performance. In several instances, model fidelity has been sacrificed by making simplifying assumptions in order to meet this objective. The initial system performance analysis indicates that three GPS satellites can be simulated on one Silicon Graphics processor using only 64% of the processing capacity. The hardware platform is a Silicon Graphics Challenge 10000 with 12 independent processors. Analysis data indicates that the requirement to simulate 30 satellites simultaneously can be met without issue.

Conclusions

This tool gives the Air Force a valuable asset needed to train the GPS operators for both nominal and off nominal operations.

The goal of the simulator is to produce results so realistic that a GPS satellite controller cannot distinguish between actual satellite data and simulated data. While this goal and the real-time goal seem to be competing, it is felt that a balance has been found in the GPS simulator.

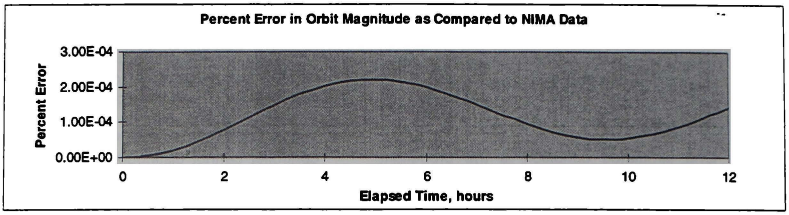


Figure 1. Predicted and Actual Ephemeris

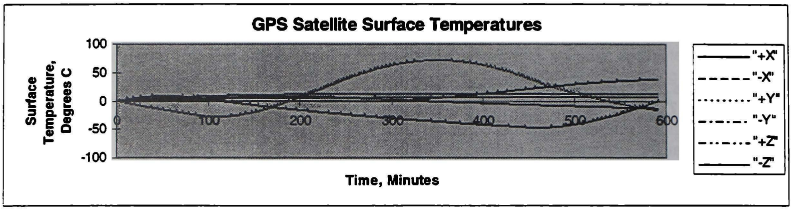


Figure 2.Satellite Surface Temperatures

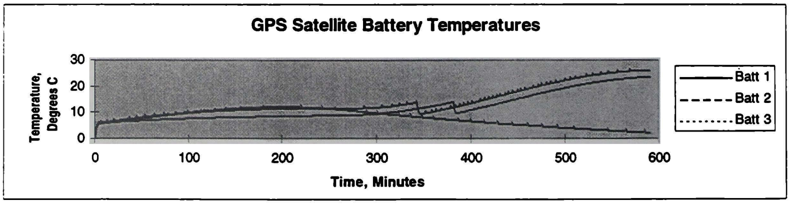


Figure 3. Battery Temperatures

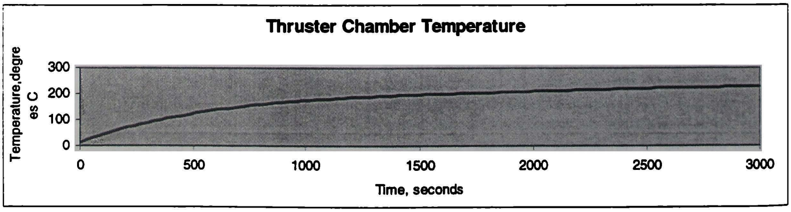


Figure 4. Nominal Thruster Chamber Temperature

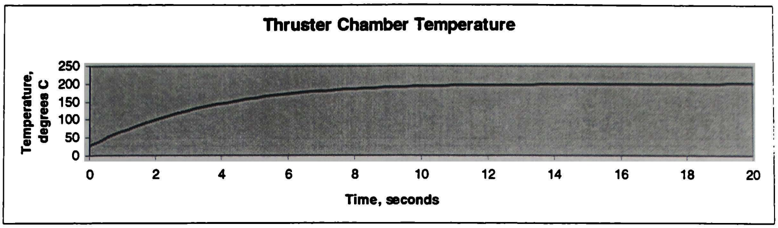


Figure 5. Thruster Chamber Temperature for Frictional Flow

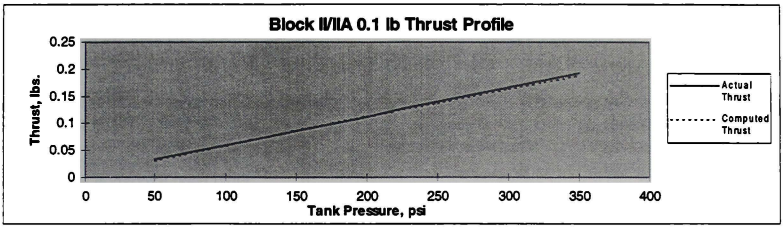


Figure 6. Thrust Comparison

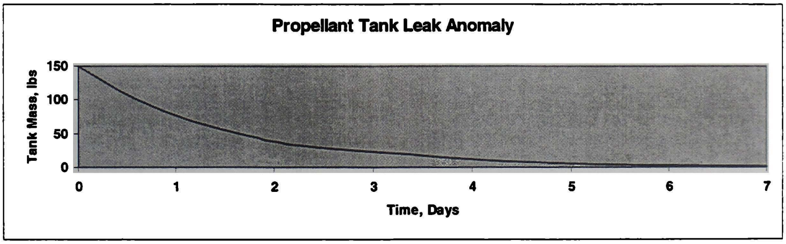


Figure 7. Propellant Tank Leak Anomaly

Constants and Variables Used in the Above Equations:

m	Mass of the surface for which the temperature is being calculated
c	Specific heat of the material for the surface
$\dot{Q}_{Joule-heating}$	The heat generated from the electrical components internal to the spacecraft
α_{SV}	Absorption factor for the surface
$A_{\perp}$	The area of the surface that is perpendicular to the Sun
K	Eclipse Factor
I_{sun}	Solar radiation energy
A_{SV}	Area of the surface that is radiating.
σ	Stephan-Boltzman's constant
ε_{SV}	Emmisivity of the surface
T_{SV}	Temperature of the spacecraft
$F_{SV-Space}$	Amount of surface that is visible to space
ε_{Space}	Emmisivity of space
T_{Space}	Temperature of space
$F_{SV-Earth}$	Amount of surface that is visible to the Earth
ε_{Earth}	Emmisivity of the Earth
T_{Earth}	Temperature of the Earth.
T_{hi}	The upper temperature limit for a particular spacecraft component
T_{lo}	The lower temperature limit for a particular spacecraft component
m	The mass of the battery.
C	The heat capacity of the battery.
η_{heater}	The efficiency of the heat transfer from the battery heater to the battery.
Q_{heater}	The thermal energy transferred From the battery heater to the battery.
$Q_{heat-of-reaction}$	The thermal energy transferred to the battery due to the chemical reaction within the battery.
$A_{battery}$	Area of the battery that is in contact with the spacecraft surface.

k_{SV}	The coefficient of thermal conductivity.
$T_{battery}$	The current battery temperature.
T_{SV}	The current temperature of the battery's mounting surface.
L	The thickness of the mounting surface.
T	Thrust
$\dot{m}$	Mass Flow Rate
V_e	Nozzle exit velocity
p_e	Exhaust pressure at nozzle exit
p_a	Ambient pressure
A_e	Nozzle exit area
k	Ratio of specific heats for the propellant
T_c	Combustion chamber temperature
p_c	Combustion chamber pressure
A_t	Nozzle throat area
R_{gas}	Universal Gas Constant
h	Fluid Heat Exchange Parameter
p	Constant derived from empirical data
q	Constant derived from empirical data
m	Mass of the thruster chamber
c	Specific heat capacity of the thrust chamber
$\dot{Q}_{Catbed-Heater}$	Thermal energy transferred from the catbed heaters to the thruster chamber
A	Surface area of the thruster chamber
T_H	Propellant temperature
T_C	Thruster Chamber Temperature
a	Constant derived from empirical data
b	Constant derived from empirical data
P_{prop}	Propellant tank pressure
E_{T_c}	Temperature effects scaling factor
$\dot{m}_{leak}$	Mass flow due to tank leakage
m_H	Mass of the hydrazine in the tank

References

1. Baker M.A., Dingman, B.K., and Gregg, W.L., “High Fidelity GPS Satellite Simulation”, Lockheed Martin Federal Systems, Gaithersburg MD, 1997.
2. Griffin M.D., and French J.R., “Space Vehicle Design”, AIAA Inc., Washington D.C., 1991.

Touch Down Simulation of the MUSES-C Satellite for Asteroid Sampling

Kazuya Yoshida [*] and Tadashi Hiraoka [†]
Tohoku University, Sendai, JAPAN

The Institute of Space and Astronautical Science (ISAS), Japan is planning to launch a satellite to obtain samples from the surface of an asteroid then return to Earth. The satellite, named MUSES-C is targeting one of near Earth asteroids. Since the gravity of the asteroid is very small, the satellite will not be able to stand on its surface, but make an impulsive contact in a free-flying situation. The free-flying and contact dynamics model is developed to investigate the touch-down motion in sample acquisition. The contact model is verified by simplified experiments, then the dynamics simulations are carried out for feasible touch-down conditions.

Introduction

ASTEROIDS are small particles of rocky bodies orbiting the Sun, a concentration of which bodies form an asteroid belt between Mars and Jupiter. The investigation into the astronomical questions on where these bodies come from, why they concentrate there, and what materials they are composed of, brings us significant knowledge on the origin and history of our solar system. The most informative method to answer these questions is to obtain samples from these planetary bodies themselves.

The Institute of Space and Astronautical Science (ISAS), Japan has a plan to launch an exploration robot satellite, named MUSES-C, which can touch down on a surface of an asteroid and acquire samples off the surface, then take them back to the Earth.[1] In a tentative mission scenario, MUSES-C, a 400 [kg] satellite, will target one of the near earth asteroids, estimated less than 1 [km] diameter rocky object. Candidates for such asteroids are "NEREUS" and "1989ML." The MUSES-C satellite approaches to the asteroid's surface in a small relative velocity at the final moment, and makes contact using a sampling probe. Inside the probe, a projectile is projected toward the surface with some high velocity to crash it, then rebounding particles, ejecta, will be collected in the probe.

There are many technical challenges in this mission. Particularly contact with the surface of asteroid is one of the most critical challenges. In order to make sure the safety, in terms of the strength of the structure against the impulsive force and the attitude maintenance against the impulsive moment, we need to carefully design the mechanisms and control systems, and simulate their dynamic behavior with making use of our maximum knowledge on the free-flying and contact dynamics. Only limited experiments are possible to test micro-gravity environment on Earth, and hardware verification with a full-scale model is usually infeasible. Computer simulations are therefore a significant approach to study this design problem.

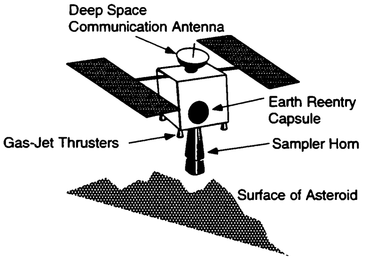


Fig. 1 The asteroid sample-return satellite, MUSES-C

This paper discusses the dynamic simulation of the touch-down sequence with the development of free-flying and contact models then, using tentative design parameters of MUSES-C, illustrate the dynamic motion after the contact.

MUSES-C Mission Scenario

MUSES-C

The MUSES-C is a satellite for the asteroid sampling-return mission which is planned by ISAS to launch in 2002. Figure 1 depicts a basic configuration of the satellite, with 400 [kg] total mass and the dimension of the main body: 1.6[m] × 1.0[m] × 1.0[m]. As appendages, the satellite has an ion engine system for interplanetary voyage, a high-gain antenna for deep space communication, solar paddles, thruster propulsion systems, the sampling mechanism called "sampler horn" and a reentry capsule back to Earth. After the sampling action, sample particles are collected and

[*] Associate Professor, The Space Robotics Lab., Dept. of Aeronautics and Space Engineering, Aoba 01, 980-8579, AIAA member.

[†] Currently works for NAMCO, JAPAN.

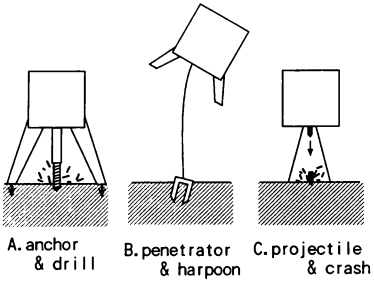


Fig. 2 Sampling Methods

packed into the reentry capsule, then its door is latched and sealed carefully to avoid contamination. When the satellite returns to Earth, only the reentry capsule which we hope filled with a lot of informative samples, parachutes down to the Earth's surface.

Asteroids

Asteroids are small particles of rocky bodies orbiting the Sun. Up to now, we have a very limited information about these small planets through telescopes, and analysis of meteors. Recently, impressive pictures of some asteroids are taken by deep space exploration satellites, such as Galileo. The pictures show that an asteroid is not a spherical planet but a very oblique and rugged rock with craters. Generally speaking, these images agree our scientific expectation in the point that, for example, the gravity is not strong enough to form a spherical planet in these size of objects. But a specific composition, if an asteroid is a huge monolith or a cluster of soft soils, if the surface is rocky, sandy or dusty... all these are open questions, and the answer should depend on the history of each asteroid.

In a tentative mission scenario, MUSES-C will target the asteroid NEREUS, or 1989ML. So far our knowledge is very limited on these asteroids, particularly its gravity and the surface condition (hardness, roughness), and right answers should be given only when the MUSES-C makes a physical contact with it. For the purpose of the simulation study, however, we need proper assumptions. Here we shall make the following assumptions as listed in Table 1.

Table 1 Characteristics of an Asteroid

gravitational acceleration [m/s^2]	:	g	9.8×10^{-4}
stiffness of surface [N/m]	:	K_w	10000
coefficient of friction	:	μ	0.5
gradient of the surface [deg]	:	θ	± 30

Sampling Strategy

To obtain samples from a small planet, which does not have enough gravity to firmly fix the exploration satellite on its surface, the following strategies depicted in Figure 2 have been discussed.

Drilling technology (A) may work effectively for the sampling from a comet, which is considered to be composed of relatively *soft* materials such as dusty, icy, and snowy compounds.

However an asteroid is considered as a more rocky or stony object covered with relatively *hard* surface, then we need more high-energy methods to crash the surface.

One of such methods is with a penetrator capsule (B). The explorer throws or projects a capsule with a tether down on the planet, and samples will be packed into the capsule by the impulsive energy of collision. In this method, an issue will be how to pull the capsule off the surface and retrieve it safely.

Currently a group of people are developing a projector method (C) and its possible designs. The basic idea is to project a 5-10 grams projectile toward the asteroid's surface with several hundreds [m/s] velocity inside the "sampling horn." We expect this provides enough energy to crash the surface and the rebounding particles, or ejecta will be collected at the top corner of the horn. Since the sampling by the projector system will complete very quickly, the exploration satellite is required to maintain the sampler contact for very short while, say 2-3 seconds, on the surface of the asteroid. This point is very favorable to our touch-down scenario.

Sampling Sequence

The gravitational force of the asteroid is very weak, estimated as one ten-thousandth of the earth gravity, the situation is therefore not that the satellite makes "landing" or "standing" on its surface but it does "rendezvous" and "berthing" in the free-flying environment. Assuming that MUSES-C takes the projection & crash method (Figure 2(C)) for sampling, we can summarize the sampling scenario as follows (see Figure 3):

1. The MUSES-C satellite makes rendezvous with the asteroid and descends to a point of interest.

2. About 5 [m] height from the surface, MUSES-C is controlled its descending velocity to zero (hovering by thrusters,) then the thrusters are turned off to freely fall down on the surface. This will result 0.1 [m/s] vertical velocity at the surface contact.

3. The sampler horn, a contact probe is compliantly mounted on the satellite main body. The compliance works to reduce the contact impulse and extend the contact period.

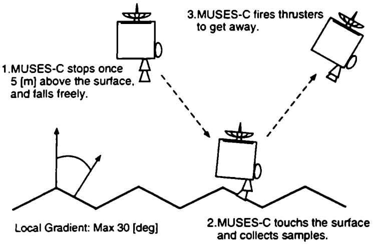

3.MUSES-C fires thrusters to get away.
1.MUSES-C stops once 5 [m] above the surface, and falls freely.
Local Gradient: Max 30 [deg]
2.MUSES-C touchs the surface and collects samples.

Fig. 3 Sampling Sequence of MUSES-C

4. While the endtip of the sampler horn stays on the surface, a projector is triggered and samples (ejecta) are collected inside the horn.

5. Thruster propulsion will follow immediately after the sampling, to get the satellite away from the surface. Note that thruster propulsion should turn on only after the sampling to prevent the contamination of the samples.

Modeling

Free-Flying Dynamics

To discuss flying or floating robot dynamics, we consider a general model that a robot satellite has plural arms including solar paddles, reaction wheels or other appendages. Such a robot satellite is modeled by a chain of free-floating links in a tree configuration consist of $n+1$ rigid bodies, connected with n articulated joints. Assume that ℓ pieces of arms are mounted on a satellite main body, and the arm k has n_k pieces of links, then $n = \sum_{k=1}^{\ell} n_k$. An example with a single arm is depicted in Figure 4.

Flexible arms or solar paddles can be treated as segmented virtual rigid links connected with elastic hinges. The flexibility yields elastic forces on the virtual hinges according to their virtual deformation. In this paper, we do discuss the compliance and deformation of the sampler horn, but do not discuss the flexibility of solar paddles or other appendages to avoid complexity.

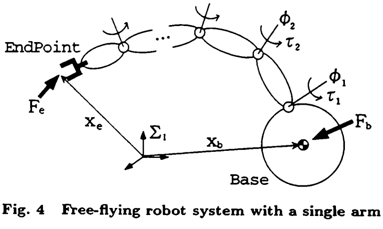


Fig. 4 Free-flying robot system with a single arm

We assume that the system freely floats in the inertial space, and no orbital motion is considered.

Let us define the following coordinates and driving forces applying on the system.

$\boldsymbol{x}_b \in R^6$: position/orientation of the base
$\boldsymbol{\phi} \in R^n$: joint angle of the arm
$\boldsymbol{x}_e \in R^6$: position/orientation of the endpoint
$\mathcal{F}_b \in R^6$: thruster force/moment on the base
$\boldsymbol{\tau} \in R^n$: joint torque of the arm
$\mathcal{F}_e \in R^6$: external force/moment on endpoint

Here we can obtain the equation of motion in the following form:[2–5]

$$\begin{bmatrix} \boldsymbol{H}_b & \boldsymbol{H}_{bm} \\ \boldsymbol{H}_{bm}^T & \boldsymbol{H}_m \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{x}}_b \\ \ddot{\boldsymbol{\phi}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{c}_m \end{bmatrix} = \begin{bmatrix} \mathcal{F}_b \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_b^T \\ \boldsymbol{J}_m^T \end{bmatrix} \mathcal{F}_e \tag{1}$$

where

$$\boldsymbol{H}_b \in R^{6\times 6} \equiv \begin{bmatrix} w\boldsymbol{E} & w\tilde{\boldsymbol{r}}_{0g}^T \\ w\tilde{\boldsymbol{r}}_{0g} & \boldsymbol{H}_\omega \end{bmatrix} \tag{2}$$

$$\boldsymbol{H}_{bm} \in R^{6\times n} \equiv \begin{bmatrix} \boldsymbol{J}_{Tw} \\ \boldsymbol{H}_{\omega\phi} \end{bmatrix} \tag{3}$$

$$\boldsymbol{H}_\omega \in R^{3\times 3} \equiv \sum_{k=1}^{\ell} \sum_{i=1}^{n_k} (\boldsymbol{I}_i^k + m_i^k \tilde{\boldsymbol{r}}_{0i}^{kT} \tilde{\boldsymbol{r}}_{0i}^k) + \boldsymbol{I}_0 \tag{4}$$

$$\boldsymbol{H}_{\omega\phi} \in R^{3\times n} \equiv \sum_{k=1}^{\ell} \sum_{i=1}^{n_k} (\boldsymbol{I}_i^k \boldsymbol{J}_{Ri}^k + m_i^k \tilde{\boldsymbol{r}}_{0i}^k \boldsymbol{J}_{Ti}^k) \tag{5}$$

$$\boldsymbol{H}_m \in R^{n\times n} \equiv \sum_{k=1}^{\ell} \sum_{i=1}^{n_k} (\boldsymbol{J}_{Ri}^{kT} \boldsymbol{I}_i^k \boldsymbol{J}_{Ri}^k + m_i^k \boldsymbol{J}_{Ti}^{kT} \boldsymbol{J}_{Ti}^k) \tag{6}$$

$$\boldsymbol{J}_{Tw} \in R^{3\times n} \equiv \sum_{k=1}^{\ell} \sum_{i=1}^{n_k} m_i^k \boldsymbol{J}_{Ti}^k / w \tag{7}$$

$$\begin{aligned} \boldsymbol{J}_{Ti}^k \in R^{3\times n} \equiv\ & [\boldsymbol{k}_1^k \times (\boldsymbol{r}_i^k - \boldsymbol{p}_1^k), \boldsymbol{k}_2^k \times (\boldsymbol{r}_i^k - \boldsymbol{p}_2^k), \ldots, \\ & \ldots, \boldsymbol{k}_i^k \times (\boldsymbol{r}_i^k - \boldsymbol{p}_i^K), \boldsymbol{0}, \ldots, \boldsymbol{0}] \end{aligned} \tag{8}$$

$$\boldsymbol{J}_{Ri}^k \in R^{3\times n} \equiv [\boldsymbol{k}_1^k, \boldsymbol{k}_2^k, \ldots, \boldsymbol{k}_i^k, \boldsymbol{0}, \ldots, \boldsymbol{0}] \tag{9}$$

$$\boldsymbol{r}_{0g} \in R^3 \equiv \boldsymbol{r}_g - \boldsymbol{r}_0 \tag{10}$$

$$\boldsymbol{r}_{0i}^k \in R^3 \equiv \boldsymbol{r}_i^k - \boldsymbol{r}_0 \tag{11}$$

m_i^k : mass of link i of arm k
w : total mass of the system ($w = \sum_{k=1}^{\ell} \sum_{i=1}^{n_k} m_i$)
$\boldsymbol{r}_i^k$: position vector of centroid of link i of arm k
$\boldsymbol{p}_i^k$: position vector of joint i of arm k
$\boldsymbol{k}_i^k$: unit vector indicating joint axis direction of link i of arm k
$\boldsymbol{r}_0$: position vector of centroid of satellite base body
$\boldsymbol{r}_g$: position vector of a total centroid of the system
$\boldsymbol{c}_b, \boldsymbol{c}_m$: velocity dependent non-linear terms
$\boldsymbol{E}$: 3×3 identity matrix

and a tilde operator stands for a cross product such that $\tilde{\boldsymbol{r}}\boldsymbol{a} \equiv \boldsymbol{r} \times \boldsymbol{a}$. All position and velocity vectors are defined with respect to the inertial reference frame.

Contact Dynamics

We assume the contact happens only at the endpoint in Figure 4 and the above formulation. Note that MUSES-C does not have what is called manipulator arm, but the endpoint of the sampler horn which is modeled as an articulated compliant arm, makes contact with an asteroid. The following discussion is how to determine the force $\mathcal{F}_e$.

In literature, there are a few papers to deal with a dynamic model of three-dimensional rigid body collision with friction. The paper by Brach[6] is one of good references. He discussed the relationship of momentum exchange and force-time product under the assumption of infinitesimal impact. A special care has been made on the treatment of frictional force, which may lead, in some conditions, physically impossible solutions called *negative energy loss*.

However, the infinitesimal impact between two of single rigid bodies is a very idealized, special case, and the Brach's algorithm cannot be directly applied to our problem. Here, we are interested in the contact between the sampler horn with compliance and the surface of an asteroid. In this case, the satellite system is not a single rigid body but an articulated multibody system which shows complex dynamic behavior. We can classify the dynamic conditions of the contact into the following three cases, and propose to take an appropriate approach to each condition.

(A) The case where all of the articulated joints are locked and the robot satellite is considered as a single rigid body during the contact, then *hard contact* happens.

- The infinitesimal impact model is true between the rigid robot satellite and the target. The Brach's algorithm can be applied directly. For this approach, we should know *the coefficient of restitution.*

(B) The case where the articulated joints are free or compliant, then *soft contact* happens.

- We should model the deformation mechanics of the compliant element and the contact surface, then track the dynamic motion of the entire articulated system. We propose a contact model later.

(C) The case where the articulated joints are not completely locked or free, but make deformation with resistance, then *small finite-time contact* happens.

- For this class of problems, the papers[7,8] give a good model, where *the virtual rotor inertia* to represent the joint characteristics of the small finite motion is used in addition to the coefficient of restitution.

Here, we will present a simplified model for the case (B), the soft contact problems.

The dynamic motion of the free-flying multibody system is described by Equation (1) with the presence of the external force $\mathcal{F}_e$. The magnitude of the force is determined by a local deformation and friction model at the contact surface.

Let us assume a point contact, then the contact moment is zero and the translational contact force $\boldsymbol{f}$ should be discussed. If we assume a model of elastic-plastic deformation in the normal (z) direction of the contact point, and Coulomb friction in the tangential directions (x and y), we have the following general expressions:

$$\begin{aligned} {}^c\boldsymbol{f}_z &= K(\boldsymbol{d})^r + D(\dot{\boldsymbol{d}})^s, && (12)\\ {}^c\boldsymbol{f}_x &\leq \mu\cos\eta\, {}^c\boldsymbol{f}_z, && (13)\\ {}^c\boldsymbol{f}_y &\leq \mu\sin\eta\, {}^c\boldsymbol{f}_z, && (14)\end{aligned}$$

where $\boldsymbol{d}$ is the depth of penetration and $\dot{\boldsymbol{d}}$ is its velocity. The left-superscript $\{\ ^c\ \}$ indicates the local coordinate frame located on the contact point. Also, μ is *the coefficient of friction* and η is the angle defined by

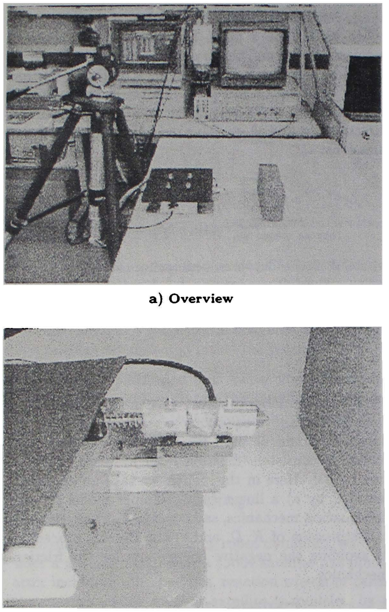
$$\tan\eta = \frac{{}^c\boldsymbol{v}_{cy}}{{}^c\boldsymbol{v}_{cx}}. \qquad (15)$$

There are number of discussions and still open questions on the above equations in the points that what numbers should be used for K, D, r and s, and how to find a consistent solution from inequality of friction model. Here in this paper, we take an approach featured by a) a linear spring-damper model for the deformation mechanics, say $r = s = 1$, b) experimental estimation of K, D, and c) the Brach's procedure of avoiding the negative energy loss for the friction mechanics.

Collision Experiment

In order to confirm the validity of the above mentioned linear spring-damper model and estimate the K, D parameters, collision experiments are carried out using an air-floating testbed. Figure 5 shows the experimental setup, and Figure 6 depicts its mathematical model. The main body that represents the satellite is an aluminum plate with $m_0 = 2.35$ [kg]. This plate is floated by pressurized air supplied through a very soft hose. On this main body, an aluminum tip with $m_1 = 0.08$ [kg] is mounted via a spring with $K_s = 630$ [N/m] and $\zeta_s = 0.3$, or $D_s = 4.2$ [Ns/m]. This part represents the sampler horn. The endtip makes contact with a wall made of firebrick, while the contact force, spring deformation, and the motion of the main body are measured and recorded.

Figure 7 (a) depicts a typical measurement of the contact force and spring deformation in the floating collision with firebrick. The measurement was carried out in 500 [Hz] frequency. The curve of the contact force has two peaks. One is relatively small and high frequency, and the other is bigger and lower frequency. Using the mathematical model of Fig.6, this is explained as a coupling vibration where the tip is bounded by two different spring-damper systems. The ratio of frequency and magnitude of two peaks depends



a) Overview

b) Close-up of the endtip

Fig. 5 Experimental setup

on the stiffness and damping of the two spring-damper systems. We know K_s and D_s and the frequency and magnitude ratio of the peaks, then the parameters of the firebrick is estimated as K_w = 10000 [N/m] and ζ_w = 0.3, or D_w = 17.0 [Ns/m]. Figure 7 (b) is a simulation using the above estimated parameters, and shows a good agreement with the experimental data (a). This fact suggests that the contact model we chose (simple linear spring-damper model), and the parameters we estimated, are good to describe the physical observation.

Simulation

Model Parameters

Figure 8 depicts a drawing of MUSES-C used for the simulation. The kinematic and dynamic parameters of main components are listed in Table 2.

The sampler horn is assumed compliant in vertical (longitudinal) direction, but constraint in other directions.

The attaching point of the sampler horn is far away from the centroid of the main body. This off-axial at-

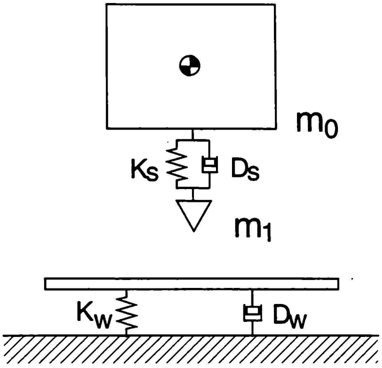
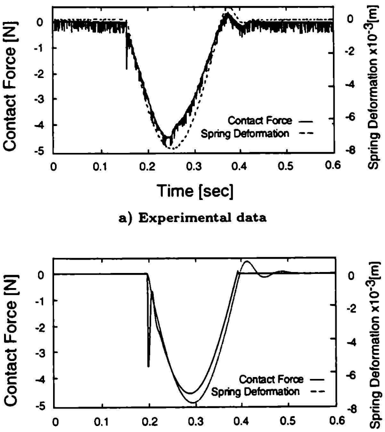


Fig. 6 Mathematical model of the experimental setup



a) Experimental data

b) Simulation

Fig. 7 Contact force-deformation profile of a floating collision experiment and the corresponding simulation

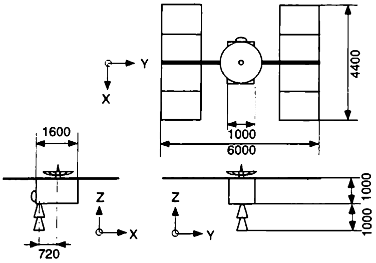


Fig. 8 A drawing of MUSES-C used for simulation. Note that this is not a final configuration, which is currently under discussion as of June 1998.

Table 2 Simulation Parameters

mass of main body [kg]	m_0	409
moment of inertia [kg·m^2]	$I_{0_{xx}}$	300
	$I_{0_{yy}}$	230
	$I_{0_{zz}}$	430
attaching point of the sampler	d_x	-0.72
horn from the centroid	d_y	0.02
of main body [m]	d_z	-0.50
mass of sampler horn [kg]	m_1	1.0
moment of inertia [kg·m^2]	$I_{1_{xx}}$	1.0
	$I_{1_{yy}}$	1.0
	$I_{1_{zz}}$	1.0
compliance of the horn [N/m]	K_s	100
damping of the horn [Ns/m]	D_s	4.3
compliance of the asteroid [N/m]	K_w	10000
damping of the asteroid [Ns/m]	D_w	17.0
friction coefficient	μ	0.5
inclination of the surface [deg]	*theta*	0

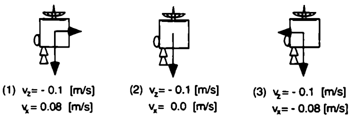


Fig. 9 Three cases of contact velocity

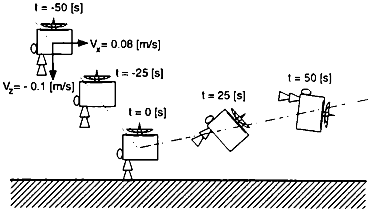


Fig. 10 Simulation results of a critical situation

tachment yields significant moment then angular motion to the main body due to the contact impulse, as will be seen in the simulation results soon. However, other design criteria will not allow to attach the horn in the center.

The surface of the asteroid is assumed with same or similar hardness and damping of firebricks. The parameters of the firebrick we identified are used in the simulation. The surface is assumed flat and horizontal.

Reaction of the Projector and Thrusters

The reaction of the projector and the gas-jet thrusters are other sources of external force on the main body than the contact impulse. The reaction of the projector is estimated to yield 3 [Nms]. Twelve of 22 [N] thrusters are mounted on the main body and four of them can be used to lift-off from the asteroid. However in the following simulation, these forces are not accounted, in order to see the nature of the physical contact and rebound.

Contact Velocity

The nominal contact velocity in vertical (z) direction is -0.1 [m/s]. The horizontal velocity, however, may be more difficult to control because the height (vertical distance) can be measured by a ranging sensor, but there is not an easy way to measure the horizontal distance. Then we set a design interface for horizontal motion control to allow plus-minus 0.08 [m/s]. In the simulation, we evaluate three cases of contact velocity as shown in Figure 9.

Contact Force and Rebounding Motion

As the results of simulation, the rebounding (lift-off) velocities, the maximum contact force f_{max}, and the contact duration time t_c is listed in Table 3.

The right column is the result without horn compliance to be compared with other three. It is clearly shown that the vertical (longitudinal) compliance in the sampler horn is very effective to reduce the contact impulse and extend the contact duration.

All results show significant rotation around y axis. This is due to the moment of the off-axial horn attachment. This rotation is very serious especially when the

Table 3 Simulation results: contact force, time and rebounding velocities

	With horn compliance			Without horn compliance
	$v_x = 0.08$ [m/s]	$v_x = 0.0$ [m/s]	$v_x = -0.08$ [m/s]	$v_x = 0.0$ [m/s]
V_x [m/s]	0.146	0.086	0.018	-0.010
V_y [m/s]	-0.004	-0.006	-0.007	-0.004
V_z [m/s]	0.032	0.066	0.093	0.064
ω_x [deg/s]	0.069	0.077	0.086	0.135
ω_y [deg/s]	1.934	1.169	0.359	1.599
ω_z [deg/s]	0.0123	-0.005	-0.021	0.017
f_{max} [N]	15.304	17.962	20.571	136.360
t_c [sec]	6.690	6.365	6.150	0.785

satellite has horizontal velocity in $+x$ direction before the contact, because it accelerates the pitch rotation. Figure 10 depicts the animated motion in such a critical case. We should carefully consider solutions to avoid the case. On the other hand, if the satellite has horizontal velocity in $-x$ direction before the contact, the moment by the off-axial horn and the moment by the horizontal velocity will cancel each other.

Conclusion

In this paper, we discussed the dynamics simulation of the MUSES-C satellite for asteroid sampling, from the free-flying and contact dynamics point of view. A mathematical model to deal with free-flying and contact dynamics is developed, and the contact model is verified by simplified experiments. Then the dynamics simulations are carried out for feasible touch-down conditions. As a result of the simulation, we find the longitudinal compliance in the sampler horn is effective, and point out a critical situation due to the off-axial attachment of the horn. We need to carefully consider the way to avoid the hazardous, and verify it by our computer simulation.

References

[1] *A Proposal of the Asteroid Sample Return Mission: MUSES-C,* The Institute of Space and Astronautical Science, 1996. (in Japanese)

[2] *Space Robotics: Dynamics and Control,* edited by Xu and Kanade, Kluwer Academic Publishers, 1993.

[3] R. Mukherjee and Y. Nakamura: "Formulation and Efficient Computation of Inverse Dynamics of Space Robots," *IEEE Trans. on Robotics and Automation,* vol.8 No.3, pp.400-406, 1992.

[4] Y. Yokokoji, T. Toyoshima, and T. Yoshikawa: "Efficient Computational Algorithms for Trajectory Control of Free-Flying Space Robots with Multiple Arms," *IEEE Trans. on Robotics and Automation,* vol.9 No.5, pp.571-580, 1993.

[5] K. Yoshida: "A General Formulation for Under-Actuated Manipulators," *Proc. 1997 IEEE/RSJ Int. Conf. on Intelligent Robots and Syatems,* pp.1651-1957, Grenoble, France, 1997.

[6] R. M. Brach: *Mechanical Impact Dynamics: Rigid Body Collisions,* John Wiley & Sons, 1991.

[7] K. Yoshida: "Impact Dynamics Representation and Control with Extended Inversed Inertia Tensor for Space Manipulators," *Robotics Research 6: the Sixth International Symposium,* pp.453-463, Hidden Valley, PA, 1993.

[8] K. Yoshida, C. Mavroidis and S. Dubowsky: "Impact Dynamics of Space Long Reach Manipulators," *Proc. 1996 IEEE Int. Conf. on Robotics and Automation,* pp.1909-1916, Minneapolis, MN, 1996.

AIAA-98-4377

A NEAR FIELD JET MODEL FOR COMPUTATION OF JET IMPINGEMENT FORCES

Pradeep Kumar[*], V. Adimurthy[†], R. Swaminathan[*], C. Unnikrishnan[*] and V. Ashok[*]
Vikram Sarabhai Space Centre, Trivandrum - 695022, INDIA

ABSTRACT

A simple and effective model for the near field of underexpanded jet has been developed for estimating the forces and moments experienced by the interstage barrel due to impingement of jet exhaust during hot stage separation of a launch vehicle. This flow field data when used in Newtonian impact theory in conjunction with confinement effect provides reasonably good estimates of forces and moments experienced by the barrel in the jet. The model provides good results as compared to wind tunnel measurements and 3-D Navier-Stokes simulation results. This model provides a fast and reliable means of estimating the forces and moments for the flight conditions even when the nozzle exit pressure ratio is high. This is inexpensive compared to actual 3D CFD simulations of the underexpanded jet..

NOMENCLATURE

D	nozzle exit diameter
L	length of barrel
M	Mach number
p	pressure
r	radius
r_b	radius of barrel
u,v	velocity components in nozzle coordinate system
x	distance from the nozzle exit plane
α	barrel inclination angle
β	flow inclination angle
γ	ratio of specific heat for jet exhaust
θ_c	nozzle half angle
ρ	density of jet
ν	Prandtl Meyer angle

Subscripts

e	nozzle exit
j	free jet
n	normal to barrel surface
o	Mach surface
*	nozzle throat
∞	ambient

INTRODUCTION

A satellite launch vehicle in general, consists of several stages and these are separated at appropriate instants along the flight trajectory. Often when the thrusting upper stage has liquid propulsion system, it is required to ensure positive acceleration to the vehicle, during the separation process. One way to ensure positive acceleration to the vehicle, is to ignite the upper stage motor a few seconds prior to the stage separation and burnout of the lower stage. The separated lower stage is pushed away by the impinging plume of the upper stage motor. In this case, it becomes essential due to performance considerations, to jettison the long interstage cylinder covering the nozzle and part of the upper stage. This second separation of the cylindrical barrel is done few seconds after the first separation of the lower stage. During the separation, the barrel moves through the billowing hot plume of the upper stage, and it is essential to estimate the forces and moments acting on the separated barrel accurately, in order to ensure collision free separation.

In order to assess these forces and moments exerted on interstage barrel by the jet plume (Fig.1), initially the Boynton[1] source flow model for the free jet in conjunction with Newtonian impact pressure on the inside barrel surface was used. These results are reported by Mahesh Kumar *et al*[2,3,4]. As it is known that Boynton's model is valid only at a large distance from the nozzle, the following attempts were made to relocate the source of the jet for making it more realistic in the vicinity of the nozzle which yielded somewhat better results.

1. Source location is such that the limiting nozzle flow stream line passes through nozzle lip.
2. Source at nozzle throat.

[*] Aerodynamics Division
[†] Aerospace Flight Dynamics Group

However, such an approach did not yield satisfactory results to be used in the design and analysis of separating bodies.

It is possible to generate the forces and moments either from CFD simulations using an NS solver or from an experimental program. Though these will provide realistic assessment of forces and moments, these are costly and time consuming. Therefore an alternative approach was sought for as reported in Ref. 5 and 6. Here a free jet solution using CFD in conjunction with Newtonian impact theory and confinement effect was used. This gave good agreement between computed pressure and forces being exerted on the barrel when compared with actual CFD solution with barrel as well with experimental results. Fig. 2 shows the pressure distribution on the barrel with full CFD solution and approach of Ref 5 and 6.

Though the prediction of forces and moments using CFD free jet data with Newtonian impact theory in conjunction with confinement effect were quite accurate (Ref. 5 and 6), the computation of CFD free jet is quite expensive though not prohibitive. Also in comparison with computation involved in the Newtonian impact theory corrected for confinement effects, the computational efforts to generate the required input of free jet data from CFD approach is quite large and thus restricting the utility of the model. As already seen, the existing jet models like Boynton model are not valid in the near field. In this paper, we propose a simple near field jet model for underexpanded supersonic jets, to be used in conjunction with Newtonian impact theory with confinement effect, which reproduces the forces and moments on the separating barrel. It is shown that the results from this model compare well with experimental and CFD data for flight and wind tunnel conditions of different pressure ratios and nozzle angles.

NEAR FIELD JET MODEL

In the case of underexpanded supersonic jets, the flow field can be divided into various zones. The outer most boundary that separates the jet exhaust gases from the ambient gases is either a curved shock (in the case of supersonic free stream) or a shear layer (if it could be termed as boundary) in the case of subsonic free stream or quiescent atmosphere. In this shear layer, there is smooth transition from high jet velocity to low ambient velocity. This shear layer will not be present in the case of inviscid analysis and can be replaced by a slip stream surface. Inside the jet boundary, there exists a barrel shock. This barrel shock is formed as compression waves formed at the intersection of expansion waves at the jet boundary coalesce. Still inside there is core in which the flowfield is unaffected by the presence of ambient atmosphere. For the present model following assumptions are made regarding the flow structure of the jet exhaust.

1. The jet exhausts into quiescent atmosphere.
2. Shear layer is neglected and thus boundary between jet exhaust gases and atmosphere is thin surface.
3. The exhaust fluid is assumed to be perfect gas with constant specific heat ratio γ.

The jet flow field is divided into three zones -- core, expansion zone and outer zone. (Fig.3)

Core Region -- In this region, the velocity field is undisturbed by the presence of atmosphere. This region is bounded by axis of symmetry on one side and Mach line emanating from the nozzle lip on the other side. In this region, it is assumed that streamlines are straight lines and are unaffected by the presence of ambient atmosphere. The axial velocity is same as that at the nozzle exit plane. This nozzle exit velocity is computed by using one-dimensional nozzle flow equation. The radial velocity is computed by finding the point from where this streamline is starting at the nozzle exit plane. The density changes in the core region occur due to mass outflow through the Mach surface. In this region, the flow field is given by

$$u(x,y) = u_e$$

$$v(x,y) = u_e\left(\frac{y\tan\theta_c}{r_e + x\tan\theta_c}\right)$$

$$\rho(x,y) = \rho_0(x)$$

$$\frac{p}{p^*} = \left(\frac{\rho(x,y)}{\rho^*}\right)^\gamma$$

Here it is assumed that the streamline is a straight line emanating from the virtual source point which is at the nozzle axis and is determined by drawing a tangent to the nozzle lip. The value of $\rho_0(x)$ is computed later using mass conservation. The position of Mach surface is computed by considering the local flow inclination and Mach angle. It is obtained by solving the differential equation

$$\frac{dy_0}{dx} = \tan\left(\beta_0 - \sin^{-1}\frac{1}{M}\right)$$

with the initial condition

$$y_0(0) = r_e$$

Here β_0 is the flow inclination angle at the Mach line and is function of x, the distance from nozzle exit plane.

Expansion Fan Region -- In this region, the flow field is essentially governed by the expansion fan lines emanating from the nozzle lip. In order to get the flow field in this region, Prandtl-Meyer expansion is used. Though, the Prandtl-Meyer expansion is valid strictly for two-dimensional flow, the axisymmetric influence is partially taken into account by considering the mass conservation. In terms of β_0, ρ_0 and M_0, the flow inclination, density and Mach number respectively at the Mach surface, one gets

$$\beta = \beta_0 + \tan^{-1}\left(\frac{y-r_e}{x}\right) - \tan^{-1}\left(\frac{y_0-r_e}{x}\right)$$

$$\nu(M) = \nu(M_0) + \beta - \beta_0$$

$$V = u_0 \frac{M}{M_0}\sqrt{\frac{1+\frac{\gamma-1}{2}M_0^2}{1+\frac{\gamma-1}{2}M^2}}$$

$$\rho(x,y) = \rho_0(x)\left(\frac{1+\frac{\gamma-1}{2}M_0^2}{1+\frac{\gamma-1}{2}M^2}\right)^{\frac{1}{\gamma-1}}$$

$$u(x,y) = V\cos(\beta)$$

$$v(x,y) = V\sin(\beta)$$

$$\frac{p}{p^*} = \left(\frac{\rho(x,y)}{\rho^*}\right)^{\gamma}$$

Here $\nu(M)$ is the Prandtl-Meyer function and is given by

$$\nu(M) = \sqrt{\frac{\gamma+1}{\gamma-1}}\tan^{-1}\left(\sqrt{\frac{\gamma-1}{\gamma+1}(M^2-1)}\right) - \tan^{-1}\left(\sqrt{M^2-1}\right)$$

For computing the density ρ_0, the principle of mass conservation is used. This gives

$$\pi r^{*2}\rho^* u^* = \int_0^{y_{lim}} 2\pi\rho u y\, dy$$

and at y_{lim} one has to satisfy the condition

$$p = p_\infty$$

Use of the last two equations provides us the values of y_{lim} and $\rho_0(x)$.

Outer Region -- In this region, the ambient conditions prevail. Thus the flow variables are given by

$$u(x,y) = 0$$

$$v(x,y) = 0$$

$$p(x,y) = p_\infty$$

$$\rho(x,y) = \rho^*\left(\frac{p(x,y)}{p^*}\right)^{\frac{1}{\gamma}}$$

Here it is tacitly assumed that the flow is inviscid and thus the boundary of the plume y_{lim} acts as slip stream across which there is no pressure differential but velocity differential exists. In reality, there will be a mixing layer in this region which allows smooth variation of velocity across the plume boundary.

It should be noted here that this is a model of jet flow field for computation of jet impingement forces. It does not account for all the flow features like barrel shock, for example.

JET IMPINGEMENT FORCE MODEL

In this approach, the free jet profile obtained either from CFD solutions or jet model is taken as the basic data. To account for the confinement of the jet due to the presence of the barrel, the density is modified and then used to estimate Newtonian pressure on the barrel surface. The new density is derived based on the condition that the mass flowing across any section of the barrel equals the mass flowing out of the nozzle throat (or exit). This is because for the cases considered here, the entire mass issuing out of the nozzle enters the barrel. The new density is evaluated by the following mass conservation relation by *assuming a density profile in the barrel* for each azimuthal plane.

$$\int_{r=0}^{r=r_j}\rho_j u r dr = \int_{r=0}^{r=r_b}\rho_{new} u r dr$$

where ρ_{new} is given by

$$\rho_{new} = \rho_j + K\left(1 - \exp\left(-n\frac{r}{r_b}\right)\right)$$

Here K denotes the change in density and n controls the density variation inside the barrel. This particular form has been chosen because it is expected that relatively large changes in the density will take place near the barrel surface and small changes at the centreline. The pressure on the barrel surface using Newtonian approximation is given by

$$p = \rho_{new} v_n^2$$

The normal force and the pitching moment are obtained by integrating the pressure on the barrel surface. As both the displacements of the barrel i.e. lateral displacement and angular displacement considered in the present analysis are in the same plane, due to symmetry considerations, the side force and yawing moment are zero. More details about the method are available in Ref. 8 and 9.

RESULTS AND DISCUSSION

The near field data from the model was computed both for the flight conditions ($p_c/p_\infty \approx 3000$, nozzle exit Mach number=4.0, γ=1.188,θ_c=10°) as well as for the wind tunnel conditions (1/100 model and $p_c/p_\infty \approx 130$, nozzle exit Mach number= 4.0, γ=1.4, θ_c=30°) . Using this data with Newtonian approximation in conjunction with confinement effect, predictions have been made for the wind tunnel conditions and flight conditions. The forces and moments have been nondimensionalized with respect to the nozzle exit dynamic pressure, with the barrel diameter as the reference length and the barrel cross section area as the reference area. The moments are calculated about the barrel C.G. which is taken to be at the centre of the barrel. At x/D = 0, the aft end of the barrel coincides with the nozzle exit plane. At y/D = 0, the barrel axis is aligned to the nozzle axis. The axial and lateral displacements are nondimensionalized with respect to the nozzle exit diameter. The forces and moments have been evaluated in the barrel coordinate system. Here, the value of y/D=0.16 and α=4° has been chosen as these represent the maximum displacements allowed for the barrel from dynamics considerations.

To validate the model, the computed forces and moments on the barrel are compared against CFD solution and experimental data. For this purpose estimates of forces and moments arrived at for a few cases from the codes UNS2D/UNS3D[7] and AS3D[8] are used. The codes UNS2D/UNS3D solve the Navier-Stokes equations using finite volume discretisation and implicit LU factorisation procedure for solving the governing difference equations. It uses body fitted algebraic grid with multi-grid option. The code AS3D uses finite volume discretisation and explicit time marching technique to simulate the flow field. It also uses Rectangular Adaptive Mesh (RAM) for grid generation. The grid is adapted to the solution as it evolves with grid enhancement and grid coarsening.

The results are also compared with data from a series of tests that were conducted on a (1/100) scale model with jet pressure ratio of around 130 (in flight it is around 3000). Forces and moments on the barrel were measured using a six component force balance[9].

Figures 4 and 5 show the variation of computed normal force and pitching moment coefficients with the movement of the barrel for α = 0° and y/D=0.16 for actual flight conditions and their comparison with CFD results. It is observed that the comparison is good. Normal force initially increases in magnitude and then settles down to a constant value. This is due to the increasing exposure of the barrel to jet flow. It is also observed that the force is negative which implies that the force tends to decrease the lateral displacement and is thus self corrective in nature. Pitching moment is also self corrective. It increases initially and then decreases in magnitude which is once again due to the increasing exposure of the barrel to the jet flow. In the beginning, only the aft portion of the barrel is exposed to higher pressure. As this area increases, there is increase in pitching moment. Once, the area ahead of CG of barrel is exposed to the flow, there is a decrease in the pitching moment.

Figures 6 and 7 show the variation of computed normal force and pitching moment coefficients with the movement of the barrel for α = 0° and y/D=0.16 for the wind tunnel simulation case. It also shows the comparison with experimental measurements and 3-D CFD result from UNS3D and AS3D codes. Here it is observed that initially there is good agreement between present predictions and the CFD data and wind tunnel measurements. However, as the barrel moves downstream, the prediction from the present model is not so good. This is possibly due to a portion of the barrel coming into jet core where density and velocity is large thus giving rise to relatively large forces. This feature was not observed in the case of flight results. It is possibly due to the large nozzle angle which gives rise to the increase in the radius of the core portion.

Figures 8 and 9 show the variation of computed normal force and pitching moment coefficients with the movement of the barrel for $\alpha = 2^{o}$ and y/D=0 and their comparison with experimental measurements and AS3D results. It is observed that the comparison is good. Figures 10 and 11 show similar results for $\alpha = 4^{o}$ and y/D=0. In both these cases with angular displacement, the agreement between present predictions and CFD results on one side and measurements on the other is very good. Figures 12 and 13 show the variation of computed normal force and pitching moment coefficients with movement of the barrel for $\alpha = 2^{o}$ and y/D=0.16 (which includes the influence of both the displacements) and the comparison with experimental measurements and AS3D results. Here it is observed that while the comparison with CFD results is good, the comparison with experimental data is not so good. This is probably due to setting errors in the experiments as the results are very sensitive to y/D. In addition to it, it is once again observed that agreement is good initially and not so good in the downstream portion.

CONCLUDING REMARKS

A simple and effective analytical model has been developed for the near field of an underexpanded jet which can be used for estimating the forces and moments experienced by the interstage barrel due to impingement of jet exhaust during hot stage separation of a launch vehicle. Comparisons with full CFD simulations for 3-D cases and with experimental measurements (for the scale model case) are good. This model is also inexpensive compared to the actual 3D CFD simulations. This model provides a fast and reliable means of generating forces and moments on the barrel for the flight conditions, which can be used for design and analysis of separating bodies.

REFERENCES

1. Boynton, F. P., Highly Underexpanded Jet Structure: Exact and Approximate Solutions, *AIAA Journal*, **5**, pp. 1703-1704, 1967

2. Mahesh Kumar and Pradeep Kumar, Jet Impingement Load on IS1/2M due to GS-2 Exhaust, Report No. GSLV-VSSC-ARD-25-110, August 1994

3. Mahesh Kumar and Pradeep Kumar, Jet Impingement Load on IS1/2M due to GS-2 Exhaust (Level II), Report No. GSLV-VSSC-ARD-25-127, Feb. 1995

4. Mahesh Kumar and Pradeep Kumar, Jet Impingement Load on IS1/2M due to GS-2 Exhaust(Level IIA), Report No. GSLV-VSSC-ARD-25-135, April 1995

5. Pradeep Kumar, Unnikrishnan, C. and Swaminathan, R., An Analytical Model for Estimating the Forces and Moments on GSLV IS1/2M during Separation, Report No. GSLV:VSSC:ARD:25:169, July 1997

6. Pradeep Kumar, Swaminathan, R., Unnikrishnan, C., Ashok, V. and Adimurthy, V., Modelling of Jet Impingement Forces during Hot Stage-Separation of a Launch Vehicle, AIAA paper 98-3156, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, July 13-15, 1998, Cleveland, Ohio

7. Unnikrishnan, C., Numerical Simulation of Flow During separation of 1/2M in the presence of GS-2 Jet, Report No. GSLV-VSSC-ARD-25-168, June 1997

8. AeroShape-3D- Technique and General Description, Computational Aerodynamics Systems Co., Moscow, 1994

9. Pandian, S., Subramaniam, S. and Kurup, M. K. A., Experimental Investigation of GS-2 Jet Load on Separating GSLV 1/2M Barrel, Report No. ATTF/11/1997, Oct. 97

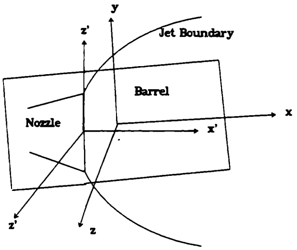


Figure 1: Schematic of Nozzle and Interstage Barrel

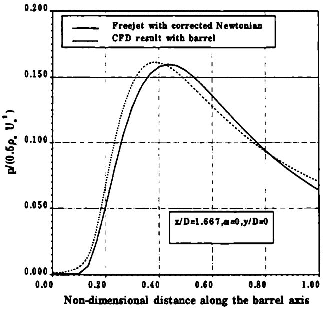


Figure 2: Pressure Distribution on Interstage Shell Surface for Axisymmetric Flow

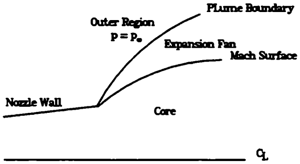


Figure 3: Schematic of Near Field Jet Model

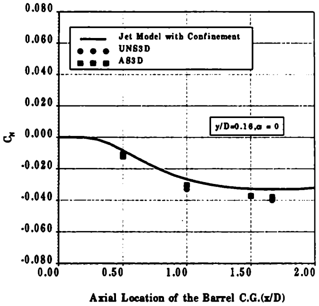


Figure 4: Normal Force Coefficient on Interstage Shell for $\alpha = 0^o$ and y/D=0.16 for Flight Conditions

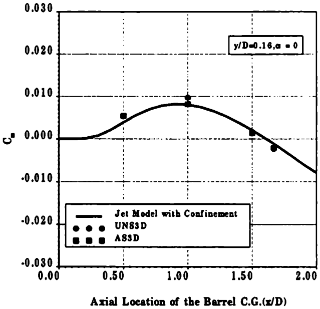


Figure 5: Pitching Moment Coefficient on Interstage Shell for $\alpha = 0^o$ and y/D=0.16 for Flight Conditions

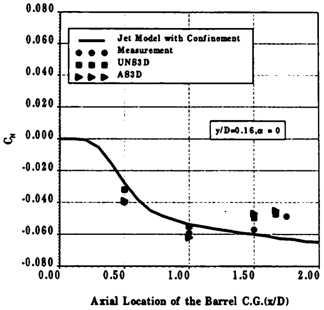


Figure 6: Normal Force Coefficient on Interstage Shell for $\alpha = 0°$ and y/D=0.16 for Wind Tunnel Conditions

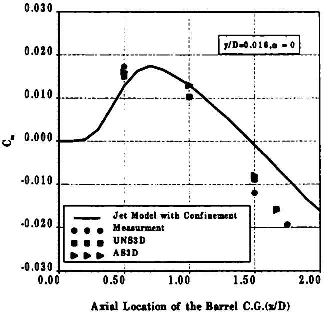


Figure 7: Pitching Moment Coefficient on Interstage Shell for $\alpha = 0°$ and y/D=0.16 for Wind Tunnel Conditions

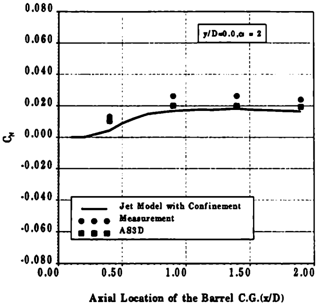


Figure 8: Normal Force Coefficient on Interstage Shell for $\alpha = 2^o$ and y/D=0.0 for Wind Tunnel Conditions

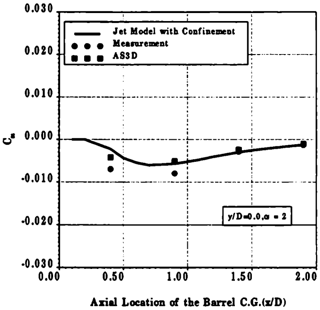


Figure 9: Pitching Moment Coefficient on Interstage Shell for $\alpha = 2^o$ and y/D=0.0 for Wind Tunnel Conditions

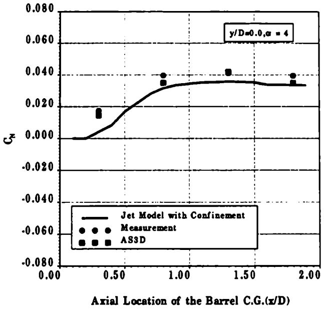


Figure 10: Normal Force Coefficient on Interstage Shell for α = 4° and y/D=0.0 for Wind Tunnel Conditions

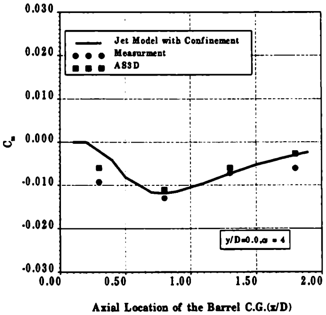


Figure 11: Pitching Moment Coefficient on Interstage Shell for α = 4° and y/D=0.0 for Wind Tunnel Conditions

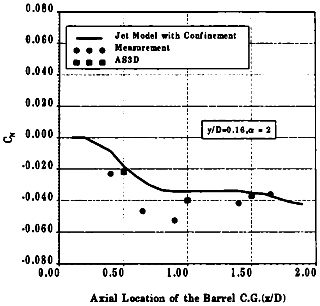


Figure 12: Normal Force Coefficient on Interstage Shell for α = 2° and y/D=0.16 for Wind Tunnel Conditions

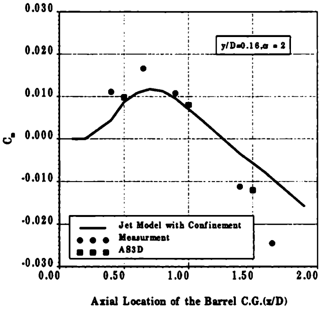


Figure 13: Pitching Moment Coefficient on Interstage Shell for α = 2° and y/D=0.16 for Wind Tunnel Conditions

AIAA-98-4527

IEEE-1394: AN EMERGING INTERCONNECTION SYSTEM FOR FUTURE SIMULATIONS

George J. Valentino
Senior Technical Manager - Innovative Systems Group
SYSTRAN Corporation
Dayton, OH 45432

ABSTRACT

The IEEE-1394 Standard for High Performance Serial Bus, and several extensions and suggested extensions to the basic standard, will for the basis for this paper. IEEE-1394, sometimes referred to as FireWire (the name coined for it by Apple Computer) is now being implemented in various devices, with chipsets, connectors, analyzers, and other peripheral items continually being released by many companies. IEEE-1394 will have a significant place in the consumer and prosumer markets, and the economies of scale of being in these markets will make IEEE-1394 an attractive technology for the simulation community. Two specific features of IEEE-1394 that will appeal to the simulation community include: (a) the capability to simultaneously support both asynchronous (guaranteed data) and isochronous (guaranteed in time) data transport services and (b) plug-and-play interconnection capability. In addition, the availability of 100, 200, and 400 Mbit/sec bandwidth capability (with higher bandwidths being defined) will provide real-time audio and video multimedia capability. This paper will examine the state of IEEE-1394 at the time of presentation (August 1998), including the salient features of the approved standards, the evolution of the standard for future features, the players within the 1394 Trade Association, available products that utilize IEEE-1394, and a vision for using IEEE-1394 to support real-time simulations.

1394 AT A GLANCE

Development of the 1394 High Performance Serial Bus can be traced to its origins as FireWire by Apple Computer Corp. in the 1980's. In fact, a significant number of real-time, 1394-equipped consumer devices have been sold by toy companies, like Toys-R-Us, for several years. I refer, of course, to the Nintendo GameBoy, pocket video game. Each GameBoy is equipped with a 1394 interface, allowing our sons and daughters to battle aliens, stack blocks, and interface with one another with one of the most capable and up-and-coming interconnection mechanisms yet devised.

Figure 1 lists some of the more compelling reasons why IEEE-1394 will be used in multiple domains, both the video game and home PC markets, as well as the prosumer home theater market and the whole house networking market. An examination of the reasons listed in Figure 1 will also show why 1394 makes sense for the real-time simulation application domain as well.

(1) Memory-mapped I/O, with 64-bit addressing
(2) 64 isochronous channels
(3) IEEE-1212 std CSR (Control and Status) register set
(4) Layered architecture for long life
(5) Peer-to-peer connectivity
(6) Bus management
(7) 100, 200, and 400 Mbits/sec (going higher)
(8) Unsupervised cabling
(9) Economies of scale via infusion into consumer market
(10) Planned upgrades for longevity

Figure 1 Compelling reasons for 1394

Memory-mapped I/O, with 64-bit addressing refers to how inputs and outputs are handled (direct I/O to/from system memory to reduce latency and allow the CPU to operate on the content of the memory); and, 64-bit addressing refers to an address size of 2^64.

Isochronous channels refers to one of two transport mechanisms in 1394. Isochronous channels provide real-time delivery services, while asynchronous transport provides guaranteed delivery of data.

Use of the IEEE-1212 within 1394 refers to the constructs used for the Control and Status registers, with 1212 being another IEEE standard.

A layered architecture for long life refers to a separation of the physical, link, and management layers so that newer implementation of each can be inserted within a design without changing the overall design of the interface.

Peer to peer connectivity means that there are no "master" or "slave" devices explicitly required for 1394 interconnections.

Bus management refers to the incorporation of bus management features within the 1394 specification that interact with the physical, link, and transaction layers for such things as configuration and error control.

The available and emerging data rates of 100, 200 and 400 Mbits/sec means that 1394 can accommodate a variety of bandwidths (12.5, 25, and 50 MBytes/sec in this case) well suited to the movement of content-rich multimedia between interconnected nodes. [Note: 1394 committees are already looking at 800, 1,600, and 3,200 Mbits/sec 1394 standards.]

Unsupervised cabling refers to 1394's ability to provide automatic address selection, with no terminations, and with arbitrary device locations. .

Economies of scale refers to the significantly sized market in-place and forecast for 1394.

Planned upgrades refers to the draft specifications within committees for high data rates, longer interconnect distances, and additional capabilities planned for 1394.

Figure 2 lists the stated (market) and actual (technical) bandwidths of the baseline and emerging 1394 standards.

Marketing	vs.	Technical Details	
• 100 Mbits/sec	=>	98.304 Mbits/sec	
• 200	=>	196.608	
• 400	=>	393.216	
• 800	=>	786.432	p1394b
• 1600	=>	1572.684	p1394b
• 3200	=>	3145.728	p1394b

Figure 2 Defined and Planned 1394 Bandwidths

Figure 3 lists the two transport mechanisms within 1394. If you recall nothing else from this paper, recall that isochronous and asynchronous transport mechanisms are defined within 1394 and that these two transport mechanisms are a kety to the promise of 1394 for the real-time delivery of multimedia content. This content can be either that included within a video game for a seven-year-old, generated during a multiple player exercise with simulated tanks and helicopters.

As the figure states, asynchronous transport guarantees delivery of data, while isochronous transport insures the data will be delivered on time.

Asynchronous vs. Isochronous Transport

- Asynchronous transport
  - "Guaranteed delivery"
  - Reliability more important than timing
  - Retries are OK
- Isochronous transport
  - "Guaranteed timing"
  - Late data is useless
  - Never retry
- IEEE-1394 provides both types of transport

Figure 3 1394 Transport Mechanisms

It should be noted that isochronous transport is an optional transport mechanism within a 1394 network. While optional, it is the required transport mechanism for multimedia deliver of content. For a specified 1394 bandwidth (as defined above in Figure 2), the isochronous transport mechanism can provide multiple channels for data delivery, with each channel composed of 125 µsec periods.

- Optional
  - But required for multimedia applications
- Multiple "channels" each 125 microsec "cycle" period
  - Channel count limited by available bandwidth

Figure 4 Isochronous Transport

Figure 5 illustrates how both types of transport mechanisms can coexist using the 125 µsec period of data transmission. In this example, three isochronous channels are used with packets for channels A, B, and C. Since Cycle M has additional time available within the 125 µsec period, then asynchronous packets A' and B', and there corresponding acknowledgments Ack A' and Ack B' can be added to complete and fill-up the available time within cycle M.

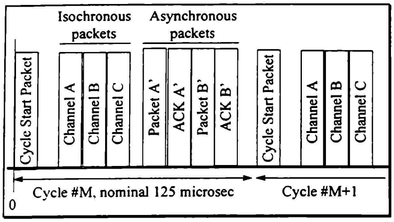


Figure 5 Cycles and Packets

As different nodes on a 1394 network generate more (or less) isochronous packets, the remaining capacity within each 125 µsec cycle can be used for less (or more) asynchronous data.

Another feature of 1394 that makes it appealing to both the consumer and simulation engineer is the concept of unsupervised cabling. Figure 6 illustrates both supervised and unsupervised cabling. With supervised cabling, specific node addresses must be associated with each source or sink on the network. Special terminators are also required. Readers familiar with the inner working of personal computer peripherals will recall that that specific node addresses must be set or encoded in such a way that the CPU "know" the location (address) of all peripherals during system initialization. The same supervised cabling paradigm is used in the instrumentation field, when devices are connected via the HP GPIB (Hewlett-Packard General Purpose Interface Bus, aka IEE-488).

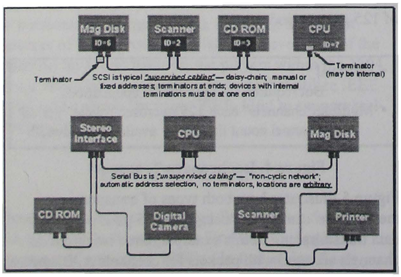

Mag Disk
ID=6
Scanner
ID=2
CD ROM
ID=3
CPU
ID=7
Terminator
Terminator (may be internal)
SCSI is typical "supervised cabling"— daisy-chain; manual or fixed addresses; terminators at ends; devices with internal terminations must be at one end
Stereo Interface
CPU
Mag Disk
Serial Bus is "unsupervised cabling"— "non-cyclic network"; automatic address selection, no terminators, locations are arbitrary
CD ROM
Digital Camera
Scanner
Printer

Figure 6 Supervised and Non-Supervised Cabling

The bottom portion of Figure 6 illustrates the concept of unsupervised cabling. In this cabling method, all node locations are arbitrary, addressing is automatic, and network terminators are not used.

A simple depiction of the 1394 protocol stack is illustrated in Figure 7. The three elements of this protocol include the physical layer (or PHY), the Link layer, and the Bus Management layer.

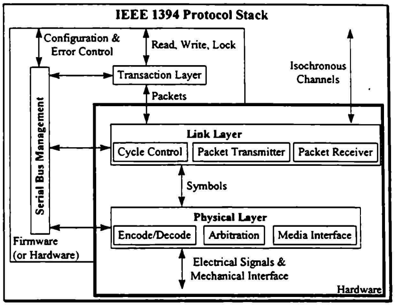


Figure 7 The 1394 Protocol Stack

The PHY is always implemented in hardware, and is the layer that connects the 1394 node to the actual electrical connector used to make the PHY or physical connection to other nodes.

The Link layer is just above the PHY, is also implemented in hardware, and serves as the transmitter and receiver of both isochronous and asynchronous packets. Isochronous packets are passed directly between the Link layer and the physical memory within the node, thus allowing low-latency, guaranteed timing exchanges between 1394 sources and sinks.

In contract to how isochronous packets are handled, asynchronous packets are passed to a hardware or firmware Bus Manager which, among other things, arbitrates packets and acknowledgments between nodes.

Figure 8 illustrates two nodes being interfaced with 1394. Each node, A and B, includes "n" 1394 ports, with "n=3" being typical. The PHY interface, connectors, and cable(s) are at the lowest-level in this interconnection scheme. By using "n=3" (or more), systems can be configured in both linear and tree configurations.

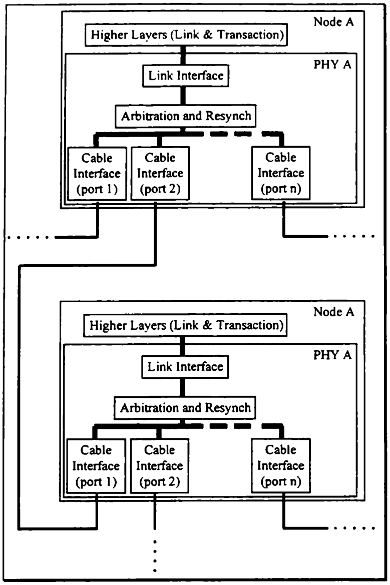


Figure 8 Interconnecting two 1394 Nodes (Note: fix Node nomenclature: A and B)

Figure 9 lists common attributes of 1394 cabling and power distribution. The typical 1394 cable is a six-conductor cable with two twisted pairs for data transport and synchronization, and one pair for power distribution. The 1394 standard allows any device to be either a source or a sink of power. There are also four-conductor cables, such as might originate in a battery-powered camera, that eliminates the power distribution cables.

The 1394 connectors, available for a multitude of sources, have been routinely used for years, again in the Nintendo GameBoy video game. These connectors are made so that the female portion of the connector, located within the node, will outlast the male portion of the plug, located on the wire. Both the connector and the plug are made to withstand hundreds of thousands of connection "make-and-breaks", a reliability required by the "it-must-always-work" consumer market. The connector is illustrated in Figure 10.

- 3-pair shielded cable
  - 2-pairs for data transport
  - 1-pair for peripheral power
- Small and rugged connector
  - 2 sockets in the same area as a mini-DIN socket
- CMOS transciever
  - 20 mv differential / 4 ma drive
- At 400 Mbits/sec, can go 4.5 m (~15 feet)
- p1394b encoding allows 800 Mbits/sec on the same media
  - perhaps even 1.6 to 3.2 Gbits/sec
- Live attach/detatch
- Up to 1.5 A (60 watts) per link
  - Nodes can either source or sink power
  - Possible to have multiple power sources on one bus

Figure 9 Cabling and Power

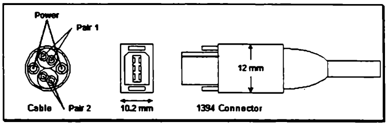


Figure 10 Connector and Cable Configurations

STANDARDS & STATUS

Figure 11 lists four standards that are directly at the heart of the 1394 movement. The first two, IEEE-1394-1995 and IEEE-1212-1991, are the defining documents for 1394. The third standard for SBP-2 involved the protocol used to interface disk drives to many PCs. Finally, IEC 1883, defines how audio and video devices, with so called Digital Video capability, can be incorporated into 1394-based interconnection systems.

- IEEE 1394-1995 High Performance Serial Bus
  - "Memory-bus-like" logical architecture
  - Serial implementation of 1212 architecture
- IEEE 1212-1991 CSR Architecture
  - Standardized addressing
  - Well-defined control and status registers
  - Standardized transactions
- X3T10 Serial Bus Protocol-2
  - SBP-2 integrates DMA into I/O process
- IEC 1883
  - IEC 1883 defines control and data for audio/video (A/V) devices

Figure 11 The Baseline 1394 Standards

For at least the last year, several committees have been investigating various embellishments to the basic 1394 standard. Figure 12 lists three emerging standards that will build upon the basic standard.

• 1394-1995	initial standard
• p1394.1	bus bridging
• p1394.a	redundant gap removal
• p1394.b	higher speeds

Figure 12 Emerging Standards

The p1394.1 standard involves "bus bridging" - allowing more than 63 devices to be installed on a single 1394 interconnect, and allowing more than 4.5 meters between nodes.

The p1394.a standard seeks to remove redundant gaps in the protocol; this will make the protocol more efficient.

The p1394.b standard is focused on adding increasingly higher bandwidth above the basic 100Mbps and 200Mbps defined in the original standard.

1394 TRADE ASSOCIATION

The 1394 High Speed Serial Bus has made significant advances during the last year and is poised to become a significant player on the desktop. One of the reasons for this advancement has been the establishment of the 1394 Trade Association. Figure 13 lists the Steering Committee members of this association. A wealth of information about this association, 1394 member companies, and product information can be found at http://www.1394ta.org.

Steering Committee of the 1394 Trade Association

Adaptec
AMD
Apple
Cirrus Logic
Compaq
IBM
Microsoft
Molex
Maxtor
NEC
Philips
Sun Microsystems
TI
Sony

Figure 13 Major Players in 1394

In July 1997, the first Development Conference for 1394 was held in San Jose. Some of the highlights of this meeting, and major points made about 1394, are listed in Figure 14. When this paper is actually presented in August 1998, I will augment the presentation with new insights gathered from the 1998 Development Conference.

Figure 15 delineates the consensus of the participates at the 1997 DevCon regarding the infusion of 1394 technology into the consumer PC. In the 1996 and 1997 period, the primary products for the PC would be add-in boards that provide 1394 interconnection capability from 100 to 400 Mbps. This projection is now reality. Several vendors, such as Adaptec and Sederta, provide such boards.

The plan for 1998 through 1999 is now unfolding. The 1394a standard has been approved and is underway.

The plan for 1999 through 2000 encompasses 1394b. 1394b seeks to increase bus speed to 1 gigabit/sec.

- Over 700 people, 46 companies, 15 countries
- 1394 is a "consumer" electronics digital interface
    - not a computer interface
- 1394 provides both asynch- and isoch-ronous transport
- 1394 is not intended to be a network
    - i.e., not a LAN, but an interconnect
- Extensions to 1394-1995 are underway
- 1394 Trade Association

Figure 14 Highlights of the 1997 1394 DEVCON

- 1996-1997
    - 1394-1995, add-in boards, motherboard components
        - 100 - 400 Mbits/sec
- 1998-1999
    - 1394A + OHCI, integrated core logic
        - 100 - 400 Mbits/sec
- 1999-2000
    - 1394B, integrated core logic
        - 100 - 400 and 800 - 1600 Mbits/sec

Figure 15 Roadmap for the PC

According to information provided in July 1997, the market for 1394 appears to be extremely significant. Figure 16 shows a market projection for the number of units across several groups of users. Here, each unit means a 1394 PHY, Link, and Transaction layer incorporated in a product. A "Pro-sumer" represents a professional individual or family that will purchase very high-end home theater products. I have added "aerospace" as a niche market outside of the mainstream 1394 application area.

• Video Post Production	10,000 units
• Videographer	100,000 units
• Videophile	1,000,000 units
• Pro-sumer	10,000,000 units
• Consumer	100,000,000 units
•	
•	
•	
• Aerospace Niche(s)	???

Figure 16 The Near Term Market

PRODUCT POPORI

As this paper goes to press, there are literally many dozens of 1394-equipped products available that span multiple market areas. A short listing of some of these products is presented here, and others will be described at the presentation of this paper in August 1998.

(1) SONY DRV-1000 DVCAM™ Player/Recorder VTR Drive – this is a compact digital video that is easily installed into a PC by fitting it into a standard 5.25" drive bay and using the internal DC supply. It is an ideal, low-cost nonlinear editor using PCs [drv1000].

(2) Adaptec AHA-8945 Combined 1394®/Ultra Wide SCSI PCI Host Adapter – allows multiple 1394 and SCSI devices to be interfaces to a host PC in a single PCI slot. 1394 transfer rates up to 200 Mbps (25 MBps) [Adaptec]

(3) The SEDNET Real-Time Interconnection Network - includes a series of host bus adapters for many bus types. Includes PMC and CPCI bus types. [Sederta]

(4) Optura Digital Camcorder - includes digital motor drive, a progressive scan CCD, and 1394 interface [Canon]

(5) DV Master - a professional video/audio nonlinear editing system [FAST]

At the Fall 1997 Comdex meeting, several companies introduced and demonstrated new 1394-capable products. Some of these are listed in Figure 17 [Comdex].

Texas Instruments introduced the industry's first 1394 Open Host Controller Interface (OHCI) device, along with a new family of 400 Megabit/ second PHY devices;

Sony, continuing its leadership in 1394-enabled consumer products, unveiled its compact new DFW-V300 color video camera, which generates images at 200 Megabits/second. Sony also populated the room with its CCM-DS250, the first digital camera to adopt the interface;

Molex, which leads the field in 1394 connectors, introduced a trio of new products, and showed the industry's first plastic optical fiber connector;

Lucent Technologies, appearing in the 1394 demo room for the first time, unveiled its six-cable transceiver/arbiter, the FW810 PHY

Symbios Logic, also a first-timer, introduced the SYM13FW403 PHY, a three-port, 400 Megabit/second implementation, along with an ATA/ATAPI-to-1394 controller that includes Microsoft's Win32 WDM driver support;

Apple appeared in the Symbios section with a joint storage demo on Mac OS using Symbios' native bridge and PCI link;

Adaptec, another of the original TA members in its third Comdex this year, provided demos of its AHA-8940 1394 PCI Host Adapter, AHA-8945 1394-Ultra Wide SCSI adapter, and the AIC-5800 1394-PCI Link Controller;

3A International, which this year co-founded the Industrial Instrumentation Work Group, showed its 1394DA-210 data analyzer and the new 1394DA-410 version, which operates at 400 Megabits/second;

Philips Semiconductors brought its 1394 Audio/Video Link Layer Controller, the PDI1394L11, along with its PHY device, used in a sophisticated digital video camera demo;

NEC's plastic optical fiber was again on display, and announced its OPCI-LINK device, which supports OHCI and interfaces with 1394 PHY layer chips that support copper wire, optical fiber and wireless transmissions.

Fujitsu Microelectronics demonstrated camera-printer interoperability based on its integrated 1394 PHY-Link product family, the MB866xx series.

Unibrain, of Athens, demonstrated its FireNet Ethernet software emulation over 1394, and showed visitors its complete FireBus road map, including FireBoard PCI host adapter, FireAttach networking set-up and other 1998 products.

Maxtor Corp. became the first disk drive company to establish a presence in the 1394 TA Demo Room.

Phoenix Technologies introduced to the Comdex audience its 1394 LINK test environment and VirtualLINK 1394a Core, which is designed for integration into PC hosts, peripherals and consumer products.

Figure 17 Fall 1997 Comdex Announcements

1394 AND REAL-TIME SIMULATIONS

It seems to this author that 1394 has a high potential for being used within the real-time simulation community. These reasons sum up my rationale for this belief:

(1) The isochronous transport mechanism within 1394, touted as the "multimedia connection" by the 1394 Trade Association, can provide the real-time delivery of data and imagery, whether it be from a kid's role playing game, or from a player defined and operating within a distributed simulation following RTI an HLA rules.

(2) Computers and peripherals with 1394 interconnects will become the norm for all consumers and business organizations within 24 months. This is a bold statement, but the market projections look right. This proliferation of such a capable interconnection mechanism will mean that real-time simulation organizations can take advantage of economies of scale and the widespread availability of 1394.

(3) Growth to gigabit/sec capacities.

(4) Low- latency -- isochronous transactions are scheduled so that they occur at 125μsec intervals.

A new paradigm, coined in the mid-1990's, is called Collaborative Engineering. This is the concept of people working and collaborating with others across facility, campus, and geographical distances. As this must by definition be an on-line, interactive process, I see it as really a variation on the global definition of real-time, distributed simulation. This Collaborative Engineering process is illustrated as the top block in Figure 18.

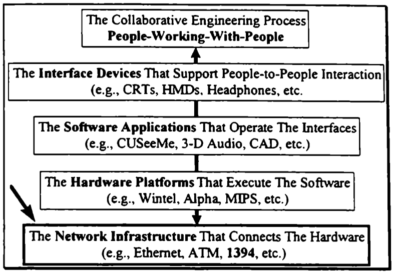

The Collaborative Engineering Process
People-Working-With-People
The Interface Devices That Support People-to-People Interaction
(e.g., CRTs, HMDs, Headphones, etc.
The Software Applications That Operate The Interfaces
(e.g., CUSeeMe, 3-D Audio, CAD, etc.)
The Hardware Platforms That Execute The Software
(e.g., Wintel, Alpha, MIPS, etc.)
The Network Infrastructure That Connects The Hardware
(e.g., Ethernet, ATM, 1394, etc.)

Figure 18 1394 in Collaborative Engineering

The three middle boxes in Figure 18 should be self-explanatory to the reader. These include the Interface Devices that actually supports the people-to-people interaction; the Software Applications that operate the interfaces (using appropriate databases, queues, and algorithms); and, the Hardware Platforms that executes the software.

At the bottom of Figure 18 is the network and interconnections mechanisms that link multiple hardware platforms. Most typical office local area networks use Ethernet, a very prolific interconnection mechanism, yet a poor choice for real-time applications. ATM (Asynchronous Transfer Mode) is another networking technology, yet one that is more typically seen in WANs (Wide Area Networks) rather than LANs. I have added 1394 to this block because of the reasons listed above (isochronous transport, economies of scale, gigabit bandwidth, low-latency).

Of course, no single interconnection technology can span the continuum, illustrated in Figure 19, from a few desktops to many geographically distributed sites. The capabilities and on-going extensions to 1394 appear to position it very well for the laboratory and facility subset of this continuum.

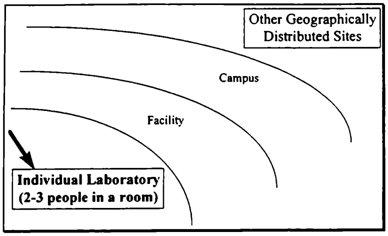

Other Geographically Distributed Sites
Campus
Facility
Individual Laboratory (2-3 people in a room)

Figure 19 1394 makes sense in Labs and Facilities

As with any new technology, its introduction must be done in phases. There is a significant infrastructure of simulation devices and simulators throughout the aerospace community. Many of these devices and simulators were design and developed years ago, yet perform well and will continue to do so for years to come. However, as the costs to operate and maintain these devices and systems escalates because of their age and uniqueness, newer devices and systems will supplant them over time.

Figure 20 illustrates a series of simulation devices at a snapshot some time in the near future. In this figure are three classes of 1394-equipped devices:

(1) COTS (Commercial Off The Shelf) 1394 devices - these will be such things as PC's (from a variety of vendors) that include multiple 1394 interfaces on the motherboard and in the device bay. You will be able to procure such devices at a variety of mainstream places such as CompUSA, Computers-R-Us, ACME Computers, and other dealers in your neighborhood.

(2) New 1394 devices - these will be the unique subsystems still required within certain simulators, but they will be developed with 1394 as the preferred high-bandwidth, isochronous, low-latency, and affordable interconnect as an integral part of its development.

(3) Legacy devices - the devices that must remain within a simulation facility because of the unique functions they perform and/or because it would be impractical to replace them (for what ever reasons). These devices will become 1394-capable via "bridges" from the devices existing interface to the 1394 interconnection mechanism.

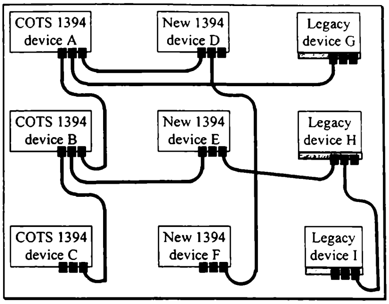


Figure 20 The "vision" of 1394 in simulation and laboratory facilities

1394 AT SYSTRAN

SYSTRAN recently started a Phase II SBIR (Small Business Innovation Research) program called B.A.C.I.S. (Binaural Audio Communications and Intercom System). This program, sponsored by AFRL/HESN at Wright-Patterson AFB, OH, will utilize IEEE-394 as the interconnection mechanism between multiple user interface nodes in this distributed digital audio system. Figure 21 provides the highlights of the B.A.C.I.S. design.

B.A.C.I.S. is being developed, as a prototype for evaluation, to provide a totally digital distribution format for audio information with the added feature that will allow individual users to "spatialize" (to place in space) the location of the audio sources one is listening to. The three major elements of B.A.C.I.S. includes the Analog Chassis, which converts existing analog signals to digital format; the SCS (Single Chassis System), which provides spatialized audio to each user, digitizes the users communications for distribution, and other user I/O functions; and the 1394 interconnection network, which provides high-bandwidth and isochronous transport capability between all B.A.C.I.S. nodes.

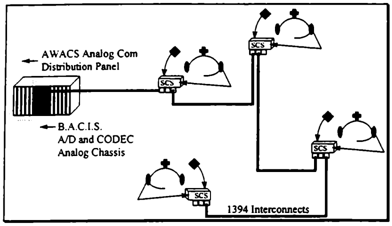

AWACS Analog Com Distribution Panel
B.A.C.I.S. A/D and CODEC Analog Chassis
SCS
SCS
SCS
SCS
1394 Interconnects

Figure 21 B.A.C.I.S. Top-Level Architecture

BIBLIOGRAPHY

[1212] IEEE Std1212-1991, IEEE Standard Control and Status Register (CSR) Architecture for Microcomputer Buses, 28 July 1992. Available from http://standards.ieee.org/.

[1394] IEEE Std 1394-1995 Standard for High Performance Serial Bus, 30 August 1996. Available from http://standards.ieee.org/.

[1394ta] Numerous documents, references, and links are available at the web site of the 1394 Trade Association at http://www. 1394ta.org

[Adaptec] http://www.adaptec.com

[Anderson 98] Don Anderson, FireWire System Architecture: IEEE 1394, 1998, ISBN 0-201-69470-0.

[Canon] http://www.canon.com

[Comdex] http://www.1394ta.org/ aboutta/comdex_products.html.

[DRV1000] http://www.sony.com

[EET_4_13_98] PC Interconnect rides on Intel's bus strategy, EE Times, 13 April 1998, p.6.

[EET_4_13_98_A] Will FireWire replace GPIB?, EE Times, 13 April 1998, p.80

[EET_5_4_98] Barriers are falling for 1394 standard, EE Times, 4 May 1998, p.49.

[FAST] http://www.fastmultimedia.com

[Fogg98] Faster FireWire: Evaluating the 1394 Interconnect for Gigabit Data Rate Transmission, HP Insite, Vol 3, Issue 2, 1998, pp.14-17.

[MSD_5_98] Report that NEC Electronics has developed prototype sample units of 1394-to-POF (Plastic Optical Fiber) repeater boxes, Multimedia Systems Design, May 1998, p.8.

[NI_S_98] Report by National Instruments, FireWire - A New World of Interface Technologies, Instrumentation Newsletter, Summer 1998, p5.

[Sederta] Home page for Sederata, source of the SEDNET "Real-time interconnection network," http://www.sederta.com.

AUTHOR INFO

GEORGE J. VALENTINO works with a group of talented individuals within SYSTRAN's Innovative Systems Group. He serves as program manager or principal investigator on a number of R&D programs with the express vision of solving specific problems while developing new hardware and software product prototypes for future commercial applications. He has an M.S. in Electrical Engineering from the Air Force Institute of Technology, Dayton, OH.

TRADEMARKS

FireWire is a registered trademark of Apple Computer Corporation.
ilink is a trademark of SONY Corp.
SEDNET is the trademark of Sederta for their 1394-based "Real-time Interconnection Network."

AIAA-98-4528

COMMERCIAL VISUAL PROGRAMMING ENVIRONMENTS: ONE STEP CLOSER TO REAL SIMULATION REUSE[*]

Allen Robins, Douglas Philbrick, Philip Bowen, Richard Klabunde
NAWCWPNS, China Lake, CA

Abstract

Simulations can be developed more efficiently when appropriate elements of existing simulations are available for reuse. In order to efficiently reuse simulation code, simulations must be built on a common framework which allows software modules to be easily shared and updated. Visual programming environments provide such a common framework. Some advantages and disadvantages of using these tools in simulation development are discussed, and lessons learned from specific examples are provided.

Introduction

The size, fidelity, and number of engineering simulations has grown to keep pace with the ever-increasing computational power of each new generation of computer. Today's laptop is as powerful as yesterday's supercomputer and certainly more accessible. The increase in simulation fidelity has generally led to larger, more complex code. We have also noticed that the number of simulation experts does not appear to be growing as fast as the number of projects requiring quality simulation support. Finally, while the work load is increasing, schedules seem to be contracting due to increased sponsor expectations and funding limitations. One effective way to tackle these problems is through the reuse of previously designed simulations and their associated software modules.

This paper addresses how the use of commercial visual programming environments is making simulation reuse practical. The paper covers three major areas. First, the benefits of simulation reuse are reviewed along with the historical barriers to achieving reuse. Second, the term "commercial visual programming environments" is defined and explained. This section also includes a discussion of the advantages and disadvantages of using these tools. Third, practical applications of how we have used these tools to become more efficient are discussed. This section briefly covers our simulation library and generic missile simulation. Also, for illustrative purposes, this paper covers the development of two medium-sized missile simulation projects. Finally, a summary of the important "lessons learned" is presented, followed by a short section of conclusions.

Benefits of Reuse

Benefits of simulation reuse are many. Of course, the most important benefits of simulation reuse come in the form of reduced time to development and of higher quality software, both of which translate directly into cost savings. Synergy amongst the community of users of common tools and models is one of the key elements to achieving this goal. This synergy manifests itself in decreased time to train new people to create, modify, and run simulations. This is partially achieved by the fact that mentoring can be done by a larger number of experts that are familiar with the "community tool set". Using common tools and models makes it easier to share knowledge and perform peer reviews. Software quality increases because testing of reusable modules tends to be more thorough, modules are reviewed by more

[*] This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

people, and effort spent on documentation is perceived as valuable.

While reuse has been a goal of our simulation community for many years, its achievement has been highly elusive. There are many historical reasons why reuse has been difficult to achieve. Traditionally, simulations have been hand-crafted by simulation experts. These experts have developed their own tool sets, making each expert individually productive. However, collectively these specialists have found it difficult to share tools and software modules. This difficulty arises because the different tools created by the various experts are rarely interchangeable. Causes include: differences in module interfaces, numerical integration techniques, analysis tools, programming languages and, of course, platform issues (SGI, HP, Macintosh, PC, etc.). Furthermore, documentation of these simulations is often terse and not up-to-date, which significantly reduces the chance for reuse. Finally, there are always the cultural issues, the resistance to change, and the "not invented here" syndrome which keeps experts from sharing each other's code.

Commercial Visual Programming Environments

Commercial visual programming environments, as they relate to engineering simulations, are typically block diagram programming environments that integrate modeling, analysis, and graphical tools into a single simulation environment. The block diagrams are hierarchical in nature which allows for the creation of complex systems from logically organized sub-system components. These window-driven software packages are currently used in many fields: aerospace, automotive, academia, etc. There are several companies which make these tools. Some of the more popular commercial tools that are available are MATLAB®/SIMULINK® from The MathWorks, Inc., MATRIX$_x$™/SystemBuild™ from Integrated Systems Inc., Easy5™ from Boeing Aerospace Co., and ACSL™ from Mitchell and Gauthier Associates.

Simulations created using these visual programming tools provide several advantages over simulations written in the traditional FORTRAN, C, and Ada programming languages. First, common software module interfaces are used within a given environment. Second, these tools provide a full time staff dedicated to product enhancement and support along with documentation and manuals. Third, commercially available toolboxes (pre-made and tested algorithms and models) are also available from industry and academia for purchase. Fourth, analysis tools are integrated into these environments which make it easier to analyze and visualize the data. Fifth, these tools allow hand coding of higher level languages where block diagramming does not make sense. Incorporation of these user written blocks also addresses reuse of legacy C and FORTRAN code. Sixth, automatic code generation from the block diagrams is often provided – instead of making diagrams to document the code, one automatically makes the code from the diagram. Seventh, visualizing the relationships between software modules is often easier using block diagrammed code than searching through traditional textual based code. Visual code makes it easier to see feedback loops and the relationships between the different sub-systems which also provides partial self documentation of the simulation. Finally, the paradigm of creating software module libraries is supported, which is strongly recommended in order to make reuse practical.

The following disadvantages to these types of tools have been found. First, these visual programming environments are currently not compatible with one another. Therefore, to achieve maximum productivity, only one of these tools should be chosen for a given organization. Second, execution speed of the completed simulation is often much slower than that of traditional hand coding due to the extra overhead. Speed penalties, however, are not as severe for block diagrams that are auto-converted to C or FORTRAN and then compiled and executed. Third, there is a significant learning curve to master these tools. This learning curve is equivalent to learning a new programming language. Fourth, there is some loss of design flexibility due to the constraints of a given tool set. For example, user-written C and FORTRAN code used in SIMULINK® must fit into a particular software template. While this requirement facilitates code reuse, it may be seen by some as a drawback. Fifth, there are risks associated with the use of tools or block diagram elements which may not have been thoroughly tested. There is a tendency to assume that commercial code is error free, which we have found to be not always true. The user is also at the mercy of the software vendor to provide updates and bug fixes in a timely manner. Finally, updating models to new versions of the tool may be painful if the vendor ceases to support or changes various features of their tool.

Practical Applications

In 1995 our group chose MATLAB®/SIMULINK® from The Mathworks as our visual programming simulation environment. Over the last three years we have used SIMULINK® in six major simulation projects and several minor ones. During this time we created a simulation library and a generic missile simulation. The simulation library contains over 20 math utilities, dealing mostly with matrix math, and five other reusable aerospace related modules: atmospheric, equations of motion, relative geometry, inertial measurement unit, and quaternion calculation blocks. The library also contains simple generic target models and a GUI-driven plotting package. The generic missile simulation was created to be the starting point for many of our other simulation projects. Each of the six major projects borrowed heavily both from the generic missile simulation and the models in the simulation library.

Some diagrams of the generic missile simulation are shown for illustrative purposes. These diagrams show some of the advantages of these types of tools. Figure 1 shows the top level of the simulation. This simulation consists of four major blocks: *Environment, Missile, Target,* and *Relative Geometry.* The wires are used to pass the data from one block to the next. The *Environment* block provides the atmospheric properties and the gravitational force for the simulation. The *Missile* block provides the missile states expressed in missile body coordinates. The *Target* block provides the inertial velocity and position states of the target. A 3 degree-of-freedom (DOF) point-mass model was used for the target, though a 6 DOF model could just as easily been used. The *Relative Geometry* block calculates the inertial position and velocity of the missile. It also calculates the inertial-to-body transformation matrices, the Euler angles of the missile, and the relative geometry between the missile and target. The *Load data* block loads the data file, initializing the model. The *Plot Buttons* block contains many pre-programmed plotting functions for a quick look at the missile and target trajectories after a simulation run.

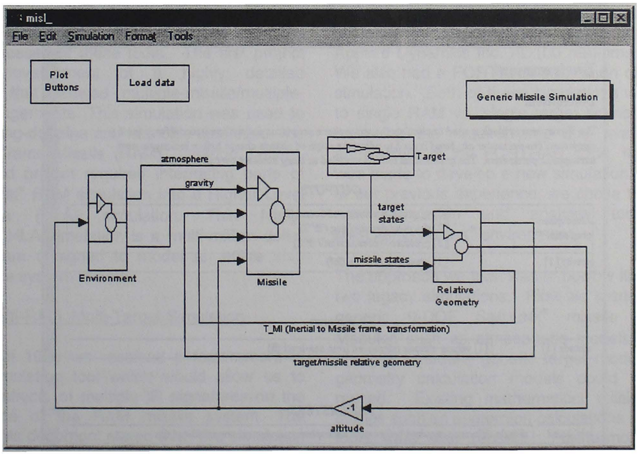

misl_
File Edit Simulation Format Tools
Plot Buttons
Load data
Generic Missile Simulation
atmosphere
gravity
Environment
Missile
Target
target states
missile states
Relative Geometry
T_MI (Inertial to Missile frame transformation)
target/missile relative geometry
-1
altitude

Figure 1 - Top Level Generic Simulation Diagram

Figure 2 shows what is "underneath" the *Environment* block. This block is accessed simply by double-clicking on the *Environment* block in the top level diagram. This block illustrates several features of these types of tools. First, the *Atmosphere* block is a user-defined block written in the C programming language. This legacy model code was placed into a special SIMULINK® template file, or wrapper, and incorporated directly into the simulation. The gravitational force is modeled simply as a constant block. The "To Workspace" block is used as a way to extract data during a simulation run. The "Subsystem Help" block at the top-right of the diagram is used to launch the web browser and load the online documentation for this environment model. The online documentation is shown in Figure 3.

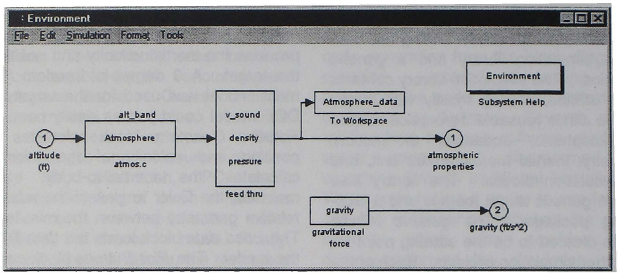
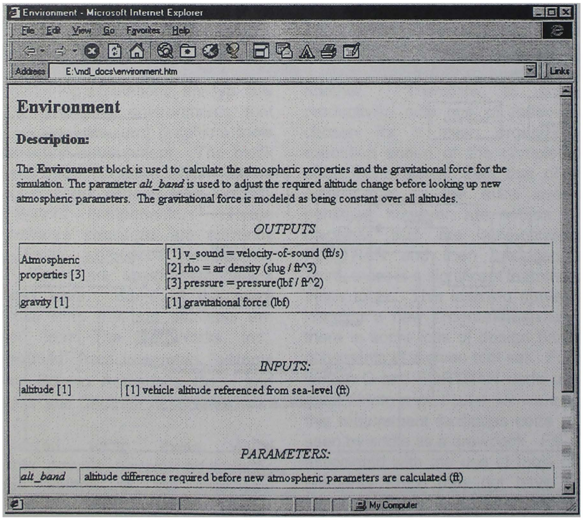


Figure 2 - Environment Model


Environment - Microsoft Internet Explorer
File Edit View Go Favorites Help
Address
E:\mdl_docs\environment.htm
Links
My Computer

Environment

Description:

The **Environment** block is used to calculate the atmospheric properties and the gravitational force for the simulation. The parameter *alt_band* is used to adjust the required altitude change before looking up new atmospheric parameters. The gravitational force is modeled as being constant over all altitudes.

OUTPUTS

Atmospheric properties [3]	[1] v_sound = velocity-of-sound (ft/s) [2] rho = air density (slug / ft^3) [3] pressure = pressure(lbf / ft^2)
gravity [1]	[1] gravitational force (lbf)

INPUTS:

altitude [1]	[1] vehicle altitude referenced from sea-level (ft)

PARAMETERS:

alt_band	altitude difference required before new atmospheric parameters are calculated (ft)

Figure 3 - Online Documentation of Environment Model

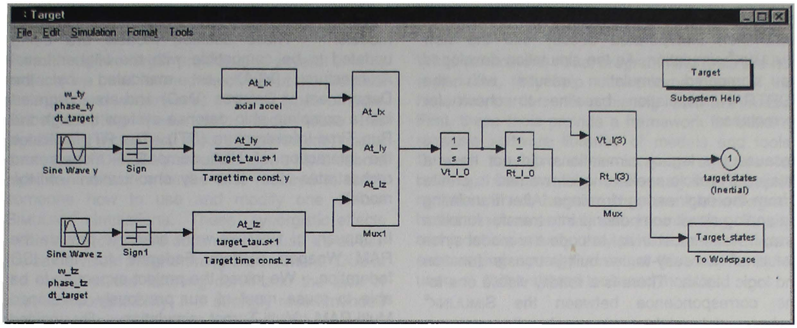


Figure 4 - Target Model

Figure 4, displayed above, shows an expansion of the *Target* block, showing some of the default blocks that come with these types of visual programming tools: sine wave generators, integrators, transfer functions, and output blocks. This simple generic target model can be used to generate many target paths: straight-and-level, corkscrews, constant g-turn, etc.

The following two subsections highlight two of the six major projects, further illustrating the strengths and weaknesses of these tools. The first project involved development of a highly detailed simulation that allowed multiple-missile/multiple-target engagements. This simulation was used to perform firing-doctrine and fratricide studies for the Rolling Airframe Missile (RAM) weapon system. The second project required integrating parts of the SIMULINK® RAM simulation into a Higher Level Architecture (HLA) simulation. This Navy-sponsored HLA simulation is a multi-million dollar pilot program designed to model an entire ship self-defense system in detail.

Multi-RAM, Multi-Target Simulation

In March of 1996 we received a requirement to build a simulation tool which would allow us to study the effects of multiple IR signatures on the performance of the RAM missile system. The requirements document stated that "this simulation will be used to evaluate how RAM plumes, warhead bursts, and other environmental features affect the overall performance of the RAM system for a variety of scenarios." Another major requirement for the simulation was that "the simulation must be configured to easily allow for either simple or complex sub-system or environmental models including a detailed IR seeker". Furthermore, there was a requirement that the simulation and its individual modules be documented and put under configuration management.

At the time of the requirement we had an accredited legacy ADSIM simulation running on an Applied Dynamics Inc. AD100 real-time computer. We also had a FORTRAN translation of the same simulation. Both of these simulations were limited to single RAM vs. single target scenarios. After reviewing the requirements and evaluating the architecture of the FORTRAN code, the decision was made to develop a new simulation. Because of our previous experience, we chose to build this new simulation and analysis tool in the MATLAB®/SIMULINK® environment.

The approach we took was to borrow liberally from two legacy simulations. First we started with our generic 6-DOF SIMULINK® missile simulation. Modules such as atmospheric models, equations of motion models, generic target models, relative geometry calculation models could be directly reused. Existing mathematical utilities, special utilities such as quaternion calculations and plotting tools were also reused. Furthermore, naming conventions and the overall simulation architecture were also retained. Second, we relied heavily on the FORTRAN legacy simulation. With little modification, we converted the non-linear aerodynamic tables from FORTRAN to C, which were then incorporated directly into the new SIMULINK® simulation. As the simulation developed we compared simulation results with the FORTRAN simulation baseline to check for correctness.

Because the legacy simulations did not have a detailed IR reticle seeker model, we had to create it from the engineering drawings. After translating the analog circuit components into transfer function form, it was almost trivial to code the model since SIMULINK® has easy-to-use built-in transfer function and logic blocks. There is a readily visible one-to-one correspondence between the SIMULINK® seeker model and the analog seeker diagrams.

One of the major drawbacks of using SIMULINK® was that in its native mode it is slow, especially for large, complex, stiff systems. With our complex seeker the simulation runs 250 times slower than real-time using a sample time of 50 microseconds. By using the Real-Time Workshop™, an add-on to SIMULINK® which provides C-code generation, we were able to convert the block diagram simulation into C code. After compiling the code, we were able to reduce the execution time by a factor of 5.

By choosing a visual programming tool such as SIMULINK® we were able to meet all of our requirements and complete the task within budget and schedule. The final product consists of detailed high-fidelity missile models which include non-linear aerodynamic tables, tabular thrust time profiles, autopilot algorithms, a simple RF seeker with associated processing electronics and a choice of a simple or highly complex IR seeker. The target models come directly from the simulation library. The simulation also has complete online documentation of each model through the use of web browser technology that is integrated into SIMULINK®.

RAM Weapon System Federate in ISD Federation

The Integrated Ship Defense (ISD) Pilot Program is a multi-laboratory, multi-contractor software integration project. One of the purposes of the project is to integrate and test the ship self-defense systems through the use of simulation. Each of the major systems is represented by high fidelity legacy models: SPS 49 radar, close-in-weapon-system (CIWS), chaff, RAM Weapon System, ship motion and the ship-self-defense-software (SSDS) models. These detailed models are being updated to be compatible with the Higher Level Architecture (HLA) as mandated by the Department of Defense (DoD) and are integrated into a complete ship defense system through the Run Time Infrastructure (RTI). The RTI facilitates the interaction of HLA compatible models and orchestrates the time synchronization of the models.

In January of 1997 we were asked to provide the RAM Weapon System Federate for this ISD federation. We joined the project expecting to be able to reuse most of our previously developed Multi-RAM, Multi-Target simulation. By making minor changes to the block diagram we were able to add the software hooks needed to exchange data between our simulation and an external source such as the RTI. Then, by using the Real-Time Workshop™ to convert the block diagram to C code, we were able to make a stand-alone executable. By modifying the main routine provided by The MathWorks Inc., we were able to easily insert and extract signals to and from this standalone simulation. The pertinent part of the modified main routine was then incorporated into an HLA wrapper that could then be plugged into the overall federation.

It is important to note that to make a simulation HLA compliant, a unique HLA wrapper is generally required for each simulation. The HLA wrapper design is dependent on the programming language chosen and the underlying simulation architecture. Architectural differences such as the input/output methodology and how the simulation advances with time greatly affect the design of the wrapper. However, by using a tool such as SIMULINK® the process is engineered just once and then applied to all other SIMULINK® simulations. Reuse of the wrapper design is possible because the language and architectural differences are removed.

Lessons Learned

From these two larger projects and several smaller projects we have learned much about the practical strengths and weaknesses of these types of tools. Since adopting these tools, code reuse has significantly increased. Also, compared with one of our legacy FORTRAN simulations, we have seen a significant reduction in the time required to teach someone how to use and modify one of our SIMULINK® simulations. These synergistic effects within our group have occurred because we all use the same visual programming environment. Due to the length of the learning curve and the lack of inter-tool compatibility, it is strongly recommended that an organization adopt only one of these tools. While large simulations may run slowly in a given environment's native mode, performance can be improved significantly by using automatic code generation. Note, however, that this is often an expensive add-on feature. Furthermore, code generation also requires that all of the user-created modules be written in the same programming language. For the most part, we are pleased with the improvements in productivity that our visual programming environment has provided. However, while doing these projects we have discovered a few new difficulties that we had not experienced previously. First, we found that although we could perform automated configuration management of the simulation through a commercial-off-the-shelf software tool, tracking specific changes to the block diagram was not easy. The problem exists because structural changes to a diagram can be made without changing the overall execution of the simulation. For instance, if one moves the position of several blocks on a page to make the diagram more readable, the version control software registers these changes as a new version of the simulation. When one then does a difference of this new version with an older version, the result is a nearly meaningless list of screen coordinate differences. The second frustration was that new software upgrades were not completely backward compatible with our older simulations. We had to recompile our user defined modules with every upgrade and on some simulations we had to make small modifications to make them work with the new environment.

Conclusions

While commercial visual programming tools are not a panacea, they help to overcome many of the historical barriers to achieving software reuse. First, these tools provide a framework for building reusable software libraries of models and tools. Second, they provide a common structure for building, documenting and analyzing complex models. Third, because they are block diagram based, they are partially self-documenting. Finally, and most importantly, they provide an environment encouraging synergy amongst the community of users of these shared tools and models.

AIAA-98-4529

LaSRS++ AN OBJECT-ORIENTED FRAMEWORK FOR REAL-TIME SIMULATION OF AIRCRAFT

Richard A. Leslie, David W. Geyer, Kevin Cunningham*
Patricia C. Glaab, P. Sean Kenney, Michael M. Madden*

Unisys Corporation
NASA Langley Research Center
MS 169
Hampton, VA 23681

Abstract

Frameworks represent a collection of classes that provide a set of services for a particular domain; a framework exports a number of individual classes and mechanisms which clients can use or adapt. This paper presents an overview of an object-oriented (OO) framework that can be used both in an interactive mode from a desktop, or to support hard, synchronous, pilot-in-the-loop real-time aircraft simulations. The abstractions used and the benefits of object technology in this environment will be discussed. The framework described in this paper has been adopted at NASA Langley Research Center (LaRC) as the basis for all future real-time aircraft simulations.

Introduction

The decision to shift the real-time simulation environment from procedural FORTRAN to OO C++ was made after serious consideration. To provide insight into this decision a brief history of the evolution of real-time simulation at LaRC over the last eight years is in order.

The legacy real-time simulation environment at LaRC was procedural FORTRAN and involved the use of several (more than six) separate software repositories. Each repository was optimized to support a specific type of research using a different simulator cockpit. Repositories were independent of each other with different variable naming conventions, COMMON block structures, or functional breakdown.

The fact that each repository was its own separate development environment made it difficult to move personnel from project to project due to the learning curve involved. Few (if any) developers fully understood all of the repositories. It was difficult to balance the workload across a limited development staff as the number of airplanes active in each repository fluctuated. Moving code from repository to repository was difficult. Significant modification and testing were required to move an aircraft model to a different repository. In the past it was felt that these costs were acceptable as development was being done on hardware that had limited memory and processing power.

In the early 1990s LaRC began to invest in multiprocessor hardware platforms with relatively large amounts of memory and computing power. As these machines became operational, discussions began about consolidating repositories in order to have a more unified environment. This resulted in a project to consolidate the FORTRAN repositories. The resulting repository was capable of meeting LaRC simulation requirements with one exception, it could not support multi-vehicle simulations. Multi-vehicle work was supported by a separate repository that used dimensioned state variables to allow up to three vehicles to be simulated simultaneously. This repository supported distributing the simulation over two CPUs. The decision to maintain a separate repository for multi-vehicle work was due to a reluctance to impose the complexity of dealing with dimensioned state variables on single vehicle projects.

* Senior Member, AIAA

In May 1994, a prototyping effort was started to explore the feasibility of using object technology to develop a single framework that would be capable of supporting all real-time aircraft simulation at LaRC. This prototype eventually evolved into the Langley Standard Real-Time Simulation in C++ (LaSRS++) framework. The decision to make LaSRS++ the standard framework was made February 1995.

Key OO Concepts Utilized In Framework

Four basic OO concepts utilized in the LaSRS++ framework are:

1. Abstraction
2. Encapsulation/Containment
3. Inheritance
4. Polymorphism

The ability to deal with layers of complexity is facilitated by the use of abstraction. The process of abstracting away complexity allows the framework developer to provide the user with a minimal but complete interface that, through abstraction, encapsulates the details of an object's behavior. While being developed, the particular details of an object's behavior must be managed. Deciding the appropriate level of abstraction is a design challenge. In developing classes for the framework the decision was made to initially provide the minimal interface to a component. The justification being that it is easier to add methods to access encapsulated data/behavior than it is to reduce an over populated interface. The provided interface to a component constitutes a contract with the users, adding a method does not affect the current users, removing a method is a violation of the implicit contract. Violation of the contract forces the user to make modification to source code that uses the modified class. If the contract remains intact, and just the internal implementation changes, then the user can recompile and relink without having to modify client source code.

The C++ language provides a means to control a users access to the implementation details and data in a class. In the LaSRS++ framework, the decision was made to make all data private. Public and protected accessors are provided to allow the users and derived classes, respectively, to access to the data. Data and methods that are implementation details are not made public thereby protecting the integrity of the data and retaining a level of control of an objects state. This prevents accidental corruption of data contained in the object. Furthermore, access through member functions allows the function that returns a value to be replaced by one that executes an algorithm and returns the result without users having to modify code. The contract remains intact.

This characteristic of containing data allows the user to created multiple instances of a given airplane without having to deal with dimensioned variables. In fact, by observing a rule forbidding data to exist outside of an object, all aircraft developed in the framework are multi-vehicle capable. If one wants multiple aircraft one just creates multiple instances of that aircraft.

Classes are the fundamental building blocks in C++. A class usually represents a single concept or physical entity. A developer specifies the behaviors (functions) and the data that a class will contain. An object is a particular instance of a class. It is useful and proper to think of classes like one thinks of intrinsic data types (int, float, double).

The two most common relationship between classes are inheritance and containment. Inheritance is often referred to as an "is a" relationship in that, when one class inherits from another there is an implicit assumption that the derived class is a either a specialization or an implementation of the base class. For example the B-757 "is a" Aircraft. Perhaps the most powerful use of inheritance is to provide abstract interfaces to subclasses. The base class can define access to behavior that is fully or completely defined in the derived classes. This mechanism, known as polymorphism, is implemented in C++ by use of virtual and pure virtual functions. The other common relationship is when a class contains a reference to a class or classes internal to itself. This is often the case when those classes are used to implement the containing class. This relationship is sometimes called "has a", containment, or aggregation.

The use of OO C++ and the enforcement of a few simple implementation guidelines within the framework create an environment where OO concepts are not just allowed, but fostered.

Framework Structure

The basic framework abstractions can be seen in Figure 1.

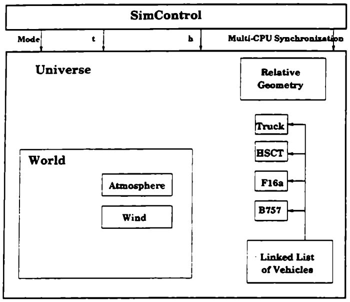


Figure 1: Framework Conceptual Diagram

All aircraft developed in the framework derive from a common set of abstract base classes that provide an interface between the framework and the specific aircraft being modeled (Figure 2).

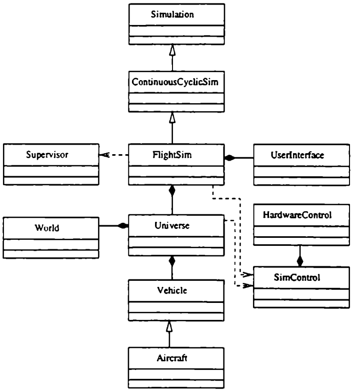


Figure 2: Top Level View Of Framework

A brief summary of the classes in Figure 2 follows:

- `Simulation`: Provides an abstract interface to allow concrete simulation types to be told to `execute()`.
- `ContinuousCyclicSim`: Adds an abstract interface that supports the concepts of mode dependent behavior and time that flows forward in fixed, discrete intervals (time step).
- `FlightSim`: Defines the initialization and execution behavior of dynamic vehicles.
- `Universe`: Provides an environment in which the vehicles execute. The concept of moving relative to a `World`, and relative geometry between vehicles are encapsulated in this class.
- `World`: Provides a world for the vehicles to fly around. `Wind` and `Atmosphere` are encapsulated here.
- `Vehicle`: An abstraction for a dynamic vehicle that has mode dependent behavior.
- `Aircraft`: Adds behavior specific to an airplane.
- `SimControl`: Provides moding and timing information in addition to multi-vehicle and multi-CPU synchronization.
- `HardwareControl`: An abstract interface to the hardware used in the simulation.
- `Supervisor`: Provides synchronization of simulation to hard real-time clock.

The modes used in the framework are defined as follows:

- reset: Used to initialize simulation/vehicles to a known state, time is set to zero.
- hold: Time does not increment, all states are frozen.
- operate: Time increments in discrete steps, vehicle dynamics are active.
- trim: Vehicles are driven to some defined steady state condition, time does not increment.

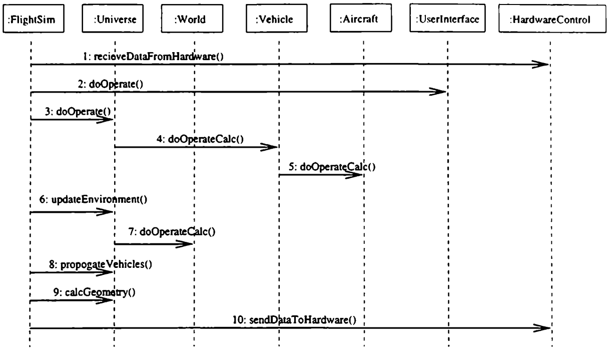


Figure 3: Top Level Object Interaction Diagram

At the highest level of abstraction the framework views all the the dynamic objects as vehicles, aircraft is just a specific kind of vehicle. Therefore, while primarily used to simulate aircraft, the framework is capable of supporting any type of dynamic vehicle. The framework provides real-time framing, mode (reset/hold/trim/operate) control, access to necessary variables, and the ability to do synchronous and asynchronous input/output to hardware. The classes describing a particular vehicle define the unique behavior that will be exhibited in a given mode.

For example, each vehicle defines its own unique behavior in operate mode(Figure 3). A brief description of each method follows:

- `HardwareControl::recieveDataFromHardware()`: Causes all synchronous and asynchronous data received from the various hardware components used in the simulation to be processed.

- `Universe::updateEnvironment()` Commands list of `World` objects to `doOperateCalc()`.

- `UserInterface::doOperate()`, `Universe::doOperate()`, `Vehicle::doOperate()`, `World::doOperateCalc()` Causes each concrete class associated with these abstract interfaces to perform whatever calculations they perform in operate.

- `Universe::propogateVehicles()` Causes the state to advance to the next time step.

- `Universe::calcGeometry()` Calculates relative geometry between vehicles.

- `HardwareControl::sendDataToHardware()` Sends the asynchronous and synchronous data generated by the simulation to hardware.

Since each vehicle defines its own behavior in the various modes (hold/trim/reset/operate), and each vehicle has slightly different components, the abstractions used are not necessarily the same across all vehicle types. However, it is useful to examine a typical abstraction for an airplane as shown in Figure 4. The interaction between the objects used to implement this airplane are shown in Figure 5.

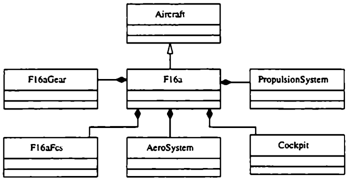


Figure 4: Typical Aircraft Architecture

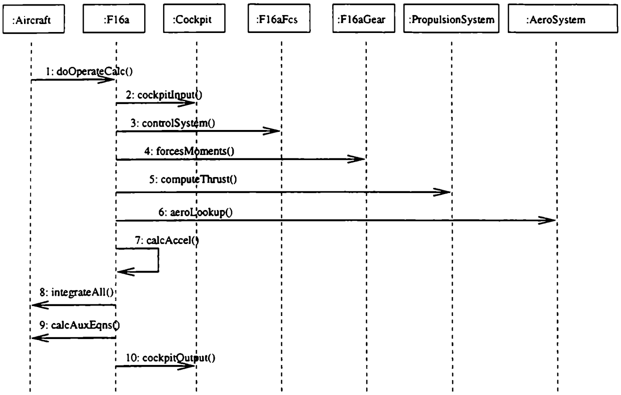


Figure 5: Typical Airplane Object Interaction Diagram

Interfaces to hardware are also abstract. Specific details of the interface to a particular piece of hardware are abstracted away and the user only has to deal with an interface that is generic. Cockpits are supported through an abstract interface that normalizes all inputs and outputs to the specific cockpit hardware.[1] While the full set of inputs needed to utilize all the capabilities of an airplane may not be present in a given cockpit, any airplane can be flown from any cockpit.

The framework is capable of supporting N-vehicles running on M-processors.[2] A current restriction is that a given vehicle's components must reside on the same CPU. Research is ongoing into the possibility of using a multi-threaded environment which would allow the distribution of a vehicle's components across CPUs.

The user interfaces with the simulation through a X-window Graphical User Interface (GUI) that provides the ability to create and delete aircraft from the simulation. Mode control, and the viewing and modification of variables in the simulation is also done through this GUI. The simulation can be operated from any X-based terminal. The same GUI interface that is used to operate production real-time is available to support checkout from the users desktop. A text based interface is also supported for use from ASCII terminals.

Due to the use of encapsulation in the simulation, the user can only modify variables through the use of methods that have been provided by the class designer in the public interface. The designer can choose to allow read-only, write-only, or read-write access to variables. Variables can also be completely encapsulated in the class and therefore protected from inappropriate modification.

In the process of developing the framework a set of "toolbox" classes were developed to provide developers with building blocks for the construction of simulation models. Toolbox functionality includes classes that:

- perform table lookups
- interface to SCRAMNet
- generate various random variate distributions
- provide both TCP/IP and TCP/UDP socket interfaces
- perform various filtering functions (such as Tustin's method)
- provide mutual exclusion.

To the extent possible the interfaces have been simplified so that users have access to capabilities that they might not have chosen to use in the past due to the complexity involved. The socket classes are an example of where users are now able to establish socket communications between distributed components in the simulations without having to deal with the details of how the interface is established.

Simulations Using Framework

To date there are four simulations supported by the framework. These simulations are a F-16A, a F-18E/F Drop Model,[3] a High Speed Civil Transport (AST-104) model and, a B-757. The B-757 is particularly interesting in that the code developed to drive experimental systems on the actual research B-757 are being developed in the simulation environment and will be moved to the aircraft with little (if any) modification. This concept called sim-to-flight has been facilitated by the use of object technology to develop the code.

Conclusions

A real-time simulation architecture has been developed utilizing OO analysis and design techniques. The architecture has been implemented using the OO programming language C++. This has resulted in a highly flexible system capable of simulating multiple instances of dynamic objects. These objects can be distributed across CPUs.

The software architecture is a self contained environment that is capable of supporting any type of dynamic object. All data is encapsulated in classes. There is no global or public data in this architecture.

Each dynamic object contains its own sequence of operations to execute. This allows each dynamic object to have uniquely defined behavior separate from other dynamic objects in the simulation. Multiple vehicles can be simulated simultaneously. The relative geometry of each vehicle relative to every other vehicle in the simulation is calculated.

The main routine driving this simulation is completely independent of both vehicle (plane, truck, ship, etc.) and simulation (continuous cyclic or discrete-event) type. This along with the supporting class library facilitates the rapid development of new types of vehicles through reuse.

Acknowledgments

The authors wish to thank the LaRC Simulation Systems Branch for their strong support in a time of change.

Bibliography

[1] P. Sean Kenney, et al. Using Abstraction To Isolate Hardware In An Object-Oriented Simulation. Paper Number AIAA-98-4533, August, 1998.

[2] Michael M. Madden, et al. Constructing A Multiple-Vehicle, Multiple-CPU Simulation Using Object-Oriented C++. Paper Number AIAA-98-4530, August, 1998.

[3] Kevin Cunningham, et al. Simulation Of A F/A-18 E/F Drop Model Using The LaSRS++ Framework. Paper Number AIAA-98-4160, August, 1998.

[4] Grady Booch. *Object-Oriented Analysis and Design*. Benjamin/Cummings, Redwood City, California, 1994.

[5] Joseph A. Kaplan, Patrick S. Kenney. SimGraph - A Flight Simulation Data Visualization Workstation. Paper Number AIAA-97-3797, August, 1997.

[6] David W. Geyer, et al. Managing Shared Memory Spaces In An Object-Oriented Real-Time Simulation. Paper Number AIAA-98-4532, August, 1998.

[7] Bruce Eckel. *Thinking in C++*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[8] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.

[9] John Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, Reading, Massachusetts, 1996.

[10] Robert C. Martin. *Designing Object-Oriented C++ Applications Using The Booch Method*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[11] Scott Meyers. *Effective C++*. Addison-Wesley, Reading, Massachusetts, second edition, 1998.

[12] Scott Meyers. *More Effective C++*. Addison-Wesley, Reading, Massachusetts, 1996.

[13] David R. Musser, Atul Saini. *STL Tutorial and Reference Guide*. Addison-Wesley, Reading, Massachusetts, 1996.

[14] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Publishing Company, Reading, Massachusetts, third edition, 1997.

[15] Patricia C. Glaab, et al. A Method To Interface Auto-Generated Code Into An Object-Oriented Simulation. Paper Number AIAA-98-4531, August, 1998.

[16] Terry Quatrani. *Visual Modeling With Rational Rose and UML*. Addison Wesley, Reading, Massachusetts, 1998.

AIAA-98-4530

CONSTRUCTING A MULTIPLE-VEHICLE, MULTIPLE-CPU SIMULATION USING OBJECT-ORIENTED C++

Michael M. Madden*, Patricia C. Glaab, Kevin Cunningham*,
P. Sean Kenney, Richard A. Leslie, David W. Geyer

Unisys Corporation
20 Research Drive
Hampton, VA 23666

Abstract

The object-oriented features of C++ simplify the design of multi-CPU, multi-vehicle simulations. Classes package data and the methods that act on the data. This packaging enables easy multiplication of objects. C++ supports inheritance and polymorphism. Polymorphism allows derived classes to redefine the methods that they inherit from their base class. Thus, client code can act on a collection of heterogeneous objects as a collection of their common base class; yet the behavior that each object exhibits is defined by its derived class type. These object features directly support the creation of heterogeneous, multi-vehicle simulations. To extend this design to multiple CPUs, developers must enable object sharing among processes or threads. Without guards, concurrent object access can lead to data corruption or program failure. This paper introduces several techniques for handling concurrent object access. Also discussed are the unique challenges to using multiple processes versus using multiple threads for multi-CPU operation. This paper uses the Langley Standard Real-Time Simulation in C++ (LaSRS++) as a successful example of applying these design techniques. LaSRS++ is an object-oriented framework for creating simulations that support multiple, heterogeneous vehicles on multiple CPUs.

Introduction

Popular procedural languages, such as FORTRAN77 and C, lead to complex designs for multi-vehicle, multi-CPU simulations. Procedural languages treat data and functions separately. The whole simulation program must have hard-coded knowledge of which functions act on which data and when. To reduce design complexity and retain maintainability, simulations using procedural languages frequently support a hard coded mixture of vehicles, fixed in their number and variety (if any). Our development team desired a single simulation framework capable of supporting any number and variety of vehicles. Such a simulation would boost productivity (through code reuse) and reduce development times.

The recent maturation of object-oriented languages, such as C++ and Smalltalk, offer a variety of techniques that bind function and data. These object-oriented features give developers new tools that simplify the design of complex systems. Object-oriented design (OOD) begins with defining classes. A *class* defines the attributes and behaviors common to a set of objects[1]. An *object* is one instance of a class.

Classes may inherit from other classes. The *derived* class inherits the attributes and behaviors of the *base* class. Derived objects extend or specialize the capabilities of their base class. Inheritance groups classes, that share common attributes and behaviors, into hierarchies. In C++, a derived class can be assigned to a

* Senior Member, AIAA

reference[†] to its base class. Clients[‡] that use the reference can only access the interface of the base class. However, *polymorphism* allows the derived class to redefine the behavior of the base class interface. Polymorphism will be discussed in more detail later.

True class design does not allow clients to directly access class attributes. Clients are only allowed to interact with the object using a functional interface. These class functions are called *methods.* Methods may not represent class attributes in the same form in which they are stored[§] nor may they expose all of the internal data in a class. This aspect of object-oriented design is called *encapsulation.*

These features of objects are instrumental in designing a simulation capable of supporting any heterogeneous collection of vehicles. Object packaging (i.e., the binding of data and function in classes) and encapsulation simplify the task of creating and using multiple vehicle objects. Inheritance and polymorphism allow the same client code to operate on different collections of heterogeneous vehicles. Having established these design principles, the discussion will address the distribution of vehicles across CPUs.

In each of these sections, the discussion introduces the design concepts as they generally apply to all objects. Then it uses examples to show how these concepts apply to vehicle simulation. The examples are drawn from the design of the Langley Standard Real-time Simulation in C++ (LaSRS++). LaSRS++ is an object-oriented framework for creating simulations. LaSRS++ is currently used at NASA Langley Research Center to support its simulation facilities. An overview of the multi-vehicle, multi-CPU design of LaSRS++ is presented in the final section. LaSRS++ is framework for building a closed-loop, continuous-cyclic simulation for real-time. In other words, the simulation breaks time into equal sized frames and executes one iteration of its event loop every frame. Although this paper discusses many general design techniques, some are appropriate only for this type of simulation. When this paper uses the term "simulation" it is referring only to this type of simulation.

The Multiplicity of Objects

One advantage of classes is that they multiply easily. Once a class is developed, it takes a single line of code to create an instance. A developer can create as many objects of the class as the hardware will support. (The number is restricted by available memory. In the case of real-time simulation, the computational speed of the hardware can also restrict the number of objects since objects must execute within the simulation's frame time). This ability demonstrates the power of encapsulation. Encapsulation allows the developer to operate on an object as a single entity even though it may internally be an aggregation of variables and other objects. Encapsulation shields the developer from dealing with the internal implementation of the object. Each object receives its own copy of class attributes. The developer does not have to manually create the desired copies of the data. For example, once a Boeing757 class has been constructed; the developer must only issue a single declaration, 'Boeing757 another_b757;', to create a Boeing757 object with all its associated data. If simulation requires another Boeing757 later, the developer simply adds another one-line declaration and another copy of data for a Boeing757 model is created.

That single line declaration does more than just create another copy of the data. It also initializes all of the data. Each class defines a special method called a *constructor.* The constructor's main purpose is to initialize all of attributes. (Constructors also create dynamically-allocated attributes.) When the constructor exists, the class has a valid initial state. In C++, constructors build on each other. A derived class automatically calls the constructor of its base class. A class constructor automatically calls all of the constructors of its attributes. This saves work for the developer.

[†] Unless stated otherwise, *reference* refers to both the reference and pointer types in C++.

[‡] Any object or function that operates on an object is a *client* of the object.

[§] To simplify discussion, *attribute* will refer to both its internal data and interface representations.

Classes also bind data with the methods that act on them. In object-oriented programming, clients never act on object attributes directly. Instead, the class presents a functional interface to the clients. This functional interface represents all of the services available to the client. A client only changes an object's attributes indirectly, by calling a method from the object's interface. In this manner, encapsulation allows the developer to selectively expose object attributes. By design, the developer can demonstrate the minimal interface required to operate the object. This is the only interface other developers need to examine; the underlying implementation is a hidden complexity.

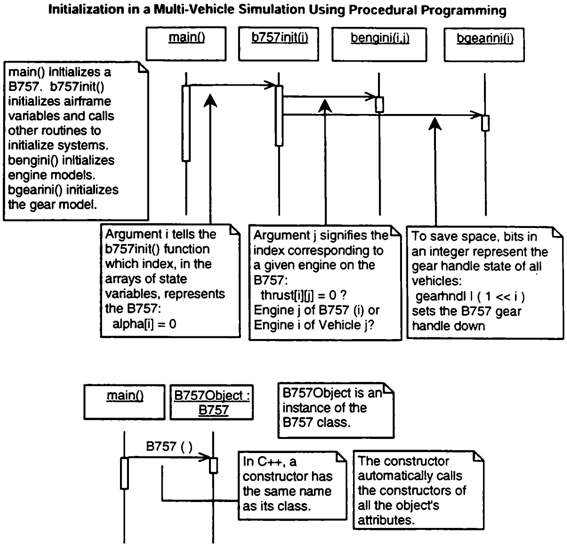

Initialization in a Multi-Vehicle Simulation Using Procedural Programming
main()
b757init(i)
bengini(i,j)
bgearini(i)
main() initializes a B757. b757init() initializes airframe variables and calls other routines to initialize systems. bengini() initializes engine models. bgearini() initializes the gear model.
Argument i tells the b757init() function which index, in the arrays of state variables, represents the B757: alpha[i] = 0
Argument j signifies the index corresponding to a given engine on the B757: thrust[i][j] = 0 ? Engine j of B757 (i) or Engine i of Vehicle j?
To save space, bits in an integer represent the gear handle state of all vehicles: gearhndl I (1 << i) sets the B757 gear handle down
main()
B757Object : B757
B757Object is an instance of the B757 class.
B757 ( )
In C++, a constructor has the same name as its class.
The constructor automatically calls the constructors of all the object's attributes.

Figure 1 Multi-Vehicle Initialization

When operating with objects, the developer does not have to track which functions operate on a particular set of data and when. The set of available actions is built into the class interface. Which of the class's attributes, that a particular method manipulates, has already been determined in the method's definition. The only step left for the developer is to invoke the action on a particular object. This binding saves bookkeeping at the design and construction phases. The developer is less concerned with designing function arguments that cause the function to operate on data representing a particular instance of a vehicle model. Clients also do not have to deal with the extra complexity of these function arguments, i.e. deciding which values are required to operate a given model. Objects provide a simple, consistent means of performing actions on then.

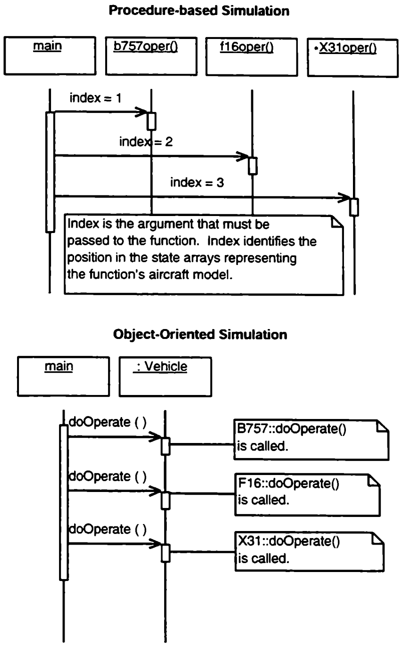
Figure 1 illustrates the differences between procedure-based and object-based simulations in implementing multiple vehicles. The figure uses a variation of the interaction diagram from the Unified Modeling Language (UML)[7]. The procedure-based simulation implements multiple vehicles using arrays (single and multi-dimensional) and packed booleans. At every level, the developer has to pass index arguments that represent the vehicle or its components (e.g. multiple engines). The developer must universally apply the same index value for a given vehicle or given component. In the engine example, the developer must know which index into a multi-dimensional array applies to an engine and which index applies to the vehicle. In the landing gear model example, the developer must also make sure that the bit position in the integer corresponds to the index position of the vehicle. In the object-based model, one constructor call initializes the object. There are no arguments that help the constructor locate the object's data. The object implicitly knows the location of its data.

Supporting Heterogeneous Vehicles

Polymorphism supports the manipulation of heterogeneous objects through a common interface. Polymorphism separates interface and behavior by allowing derived classes to change the behavior of methods declared in a base class. When a client calls a polymorphic method through a reference to the base class, the client actually invokes the behavior of the derived class. Thus, a client can act on a collection of objects through the base class interface, yet invoke behavior particular to each object. The client class makes no assumptions about the specific objects in the collection. It views the collection only as a collection of objects of the base class. The client is only coupled to the base class's interface. One can construct more derived classes, change the contents of the collection, or alter the order of the collection without the need to rewrite the client code. In C++, developers create polymorphic methods by specifying them with the *virtual* keyword. Polymorphism is invoked when a client calls the virtual method using a reference to the base class.

In LaSRS++, the Vehicle class has a virtual method called doOperate(). The F16, Boeing757, and X31 classes redefine doOperate() to execute their unique aerodynamic, engine, and flight control models. LaSRS++ stores vehicle objects on a list of Vehicle pointers. It assigns objects of all three derived classes to the list. The event loop causes all three aircraft to fly by calling the doOperate() method for each vehicle on the Vehicle list. The event loop does not contain any pre-conceived assumption about the types of vehicles in the list or their data representation. It only sees items on the list as Vehicle objects. An F18 class could later be added to the list and the client code would remain unchanged. Figure 2 illustrates the difference in multi-vehicle operation between the procedure-based and the object-based simulation.

Polymorphism simplifies the controller component of the simulation. The controller has the same responsibilities no matter what vehicle is being simulated. The controller's responsibilities include supply services for framed execution, controlling the operational state (i.e. mode) of the simulation, and managing time. By applying polymorphism to simulation design, the developer can code the controller component so that it views all vehicle models as Vehicle objects. It interacts with the vehicles using polymorphic methods introduced in the common Vehicle base class. The controller becomes capable of operating on any heterogeneous collection of vehicle models.



Index is the argument that must be passed to the function. Index identifies the position in the state arrays representing the function's aircraft model.

B757::doOperate() is called.

F16::doOperate() is called.

X31::doOperate() is called.

Running Vehicles on Multiple CPUs

So far, this paper has discussed how object-oriented programming (OOP) can facilitate the creation of a simulation that operates on a heterogeneous collection of vehicles. Without any more design considerations, this simulation is guaranteed to work only on a single CPU. Partitioning vehicles across multiple CPUs requires more planning. This discussion addresses multiple-CPU operation on a single computer. Simulations, which are distributed over multiple-computers, have other challenges, which are beyond the scope of this paper. This discussion also does not provide a complete review of all the dangers inherent in a multi-CPU programming. It focuses on problems unique to object-oriented systems.

There are two methods of running vehicles on multiple CPU's: multiple threads or multiple processes. The main difference between the two methods is memory access. Threads spawned by the same process share a common address space. Each process has its own address space, inaccessible to other processes. A common address space makes data easily sharable among threads; all that is required is a pointer or variable reference to the data. Processes, however, must transfer data through an inter-process communication (IPC) mechanism such as shared memory. Shared memory provides a means of creating a common address space among processes. Shared memory enables the developer to design a multi-process simulation similar to a multi-threaded simulation. With a little extra work, the simulation can be made portable between systems that support threads and those that do not. Because of these compelling advantages, this paper will only discuss a multi-process design that uses shared memory.

The first subsection, "Concurrent Access of Shared Objects", discusses the challenges that the two multi-CPU designs share in accessing shared objects. The two multi-CPU designs also have unique problems in sharing objects related to their memory models. These differences are the topics of the subsections "Multiple Processes" and "Multiple Threads".

Concurrent Access to Shared Objects[¶]

An object-oriented simulation designed for multi-CPU operation has a set of shared objects. More than one process operates on a *shared object*. These objects can represent frame rate, time, operational mode, world characteristics (radius, gravitational model, atmospheric model), and even the vehicle models (in combat simulations or simulations with collision avoidance). Some of these objects offer services required by all simulation models. Some of them also require that only one process invoke operations that propagate their state from one frame to the next. Whether using multiple processes with shared memory or multiple threads, restrictions must be imposed on shared objects. Otherwise, the simulation will act unpredictably or even crash

In OOP, all objects are accessed through their methods. Class methods are either mutators or constant. A mutator method modifies one or more object attributes. A constant method does not modify any object attributes. In a multi-process environment, the developer must guard against concurrent use of a mutator method with any other method. When mutator methods are concurrently invoked, they may attempt to modify the same attributes. The conflicting computations will place the object in an invalid state. When a mutator and a constant method are simultaneously active, the constant method may return an intermediate value. Both situations cause erroneous behavior or program failures.

The first task of the simulation developer is to identify the mutator and constant functions. C++ supports this division of methods. Developers can declare a method constant by tagging it with the *const* qualifier. C++ enforces the qualifier. It is a syntax error for a *const* method to modify class attributes either directly or indirectly (e.g., by calling a mutator method)[#]. The developer must then uncover those situations in which a mutator and other methods may be invoked concurrently. Before using one of the techniques described above, the developer should first examine the mutator method for possible conversion into a constant method. Developers have a tendency to make class attributes out of all the variables used by class methods. Some of these "attributes" can be converted to stack variables (i.e., local variables, method arguments, or return values). If all of the "attributes" modified by the method

[¶] For the sake of brevity, this section uses the term "process" to refer to both processes and threads.

[#] In C++, a class attribute can be qualified with the *mutable* keyword. Mutable attributes can be changed in a *const* method. The authors recommend that developers avoid the use of the *mutable* keyword for this reason. Not all compilers are equally adept at identifying situations where class attributes can be indirectly modified, particularly when data is aliased by a pointer.

can be converted into stack variables; then the developer can qualify the class as constant. This solves the concurrent access problem since concurrent execution of constant methods is always safe. Otherwise, this situation must be eliminated by using one of the techniques described below.

Developers can use several techniques to guard against concurrent use of mutators with other methods. Access to the class can be guarded using a mutual exclusion mechanism (mutex[**]). The mutex becomes an attribute of the class. Methods must first acquire the mutex before performing their function. This method guarantees that class methods will not be called concurrently. This technique is ideal for situations where client requests must be acknowledged immediately. For example, LaSRS++ contains a SharedMemory class that allows the dynamic allocation of blocks of memory. The SharedMemory class contains a semaphore. When a client calls the allocate() method, the method first attempts to acquire the semaphore before reserving a memory block of the requested size. The semaphore prevents two processes from reserving shared memory space simultaneously which can lead to overlapping blocks and corruption of shared data.

A second technique buffers attribute changes. Other processes modify the buffer and do not directly interact with class attributes. The buffer can act on behalf of one or more classes. The classes copy data in the buffer when they are ready to update their state. The buffer still requires a mutex to guarantee that its contents are not modified while the classes copy the data. However, the buffer consolidates multiple class-level synchronization actions into a much smaller number actions on the buffer. This option works well in situations where the effect of the attribute change can be deferred. Thus, this option is frequently used for user interfaces or hardware communication classes. In LaSRS++, the X-based interface changes the attributes of specific aircraft models using a buffer in shared memory. Before the aircraft updates its state, it examines the buffer for changes and copies new inputs into the appropriate attributes.

A third technique partitions program execution into blocks of events. In one event block, processes update objects they created; no interprocess communication is done. In the next block, processes communicate with each other using only constant class methods. For example, in LaSRS++, the vehicle models are executed as an event block before the relative geometry between them is calculated in the next event block. Since each process could exit the event block at a different time, the processes must communicate to each other when they are finished. This can be done with counting semaphores [1], condition variables [2], or a set of booleans (a.k.a. flags) in shared memory.

Shared flags are the simplest mechanism to implement and will be used as the example. LaSRS++ uses this technique to signal when all processes have updated their vehicles. One flag is created for each process. The flags are all initialized to false; this indicates that the processes have not updated their vehicles. As each process completes the event block, it sets its associated flag to true. Then, the process enters a *while* loop that exits only when all the flags are true.

The danger of this design is in resetting the flags. The flags cannot be reset until all processes have exited the while loop, and the flags must be reset before a process returns to the event block. Closed loop, continuous cyclic simulations have an advantage in this situation. They guarantee the following: processes synchronize at the start of frame, processes cannot overrun their frames, processes have dedicated CPUs, and processes cannot be interrupted during normal operation. These characteristics create opportunities where the flags can safely be reset. In LaSRS++, one process, designated the "main" process, resets all the flags before the frame ends. This operation is considered safe because the main process performs many functions after the vehicle update that the other "auxiliary" processes do not. Thus, the main process is always the last process to complete its event loop.

[**] Some operating systems offer a synchronization mechanism called a mutex. Mutex is used in this paper to denote any mutual exclusion mechanism including semaphores.

Division of the program into event blocks is the preferred method. It requires the least amount of work since it introduces synchronization at the program level. Synchronization does not have to be designed at the object-level or in association with buffer copies (i.e., data exchanges). Synchronization is associated only with events. Within these events, any concurrent access of objects is designed to be safe.

Unsafe concurrent access to objects is not the only factor that can compromise data integrity. Code optimization can cause similar problems. Optimizers attempt to keep as much data in the CPU's registers as possible. Optimizations may keep the value of a shared object's attribute in a register. While the attribute is in the register, the process running on that CPU does not recognize changes made to the attribute by other processes. This situation defeats careful planning to avoid unsafe concurrent access of mutator methods with constant methods. A mutator called in one process changes the attributes value; but the constant method called immediately afterward by another process continues to use the old value since it already exists in a register. Fortunately, C++ provides the type qualifier *volatile* to tell the optimizer that a variable may change in unpredictable ways and, thus, must always be read from memory[2]. All class attributes used across processes must be declared with the *volatile* qualifier.

Multiple Processes

Multiple processes share objects by placing the objects in shared memory. Using shared memory to create a common address space for the processes requires a fair amount of design work. Fortunately, C++ features seamless creation of custom allocators[††]. For each class, developers can redefine the *new* operator[‡‡]. The custom *new* operator is also inherited. If introduced in a base class, all classes in the hierarchy will use the custom new operator for dynamic allocation unless they redefine it. Shared classes can redefine their *new* operator to place them in shared memory. By redefining *new*, shared classes and non-shared classes use the same standard C++ syntax for dynamic allocation. A shared class can later revert to a non-shared class by commenting out the redefined *new* operator.

Redefinition of the new operator is almost required for custom placement of objects. Object constructors cannot be called explicitly except to create temporary objects resulting from a type conversion[3]. The *new* operator is the only means to invoke a class constructor on a dynamically allocated object. However, redefining *new* for a class causes all objects of that class to be custom allocated. This can only be overridden by explicitly invoking the global *new* operator using a scope qualifier, i.e. *::new*. If only one or two objects of a class need to be shared, C++ provides a variation of the *new* syntax called "placement syntax"[3]. Placement syntax allows the developer to specify the address where the object will be constructed. Sufficient memory to contain the object must already be reserved at that address when the "placement new" is called.

Customizing the *new* operator for shared memory allocation requires a memory manager. The memory manager creates the shared memory segment and allocates space on demand. It may reuse memory reclaimed from deleted objects. Since multiple processes could attempt to allocate or delete objects simultaneously, the memory manager must guard its allocation and deletion functions with a mutex. The developer usually has to create the memory manager; prepackaged solutions may not be available.

Since an object can contain other objects, shared objects can dynamically create other shared objects. These shared objects will then contain pointers to other locations in the shared memory segment. These pointers are valid for all processes only if all processes attach the shared memory segment using the same starting address. Fortunately, UNIX allows the developer to specify the starting address of the shared memory segment. However, the authors' experience has shown that, on some UNIX variants, it is difficult to select a

[††] An allocator reserves memory space for a dynamically created object. In C++, the operator *new* performs dynamic allocation of data.

[‡‡] The *delete* operator must also be customized. The delete operator reclaims memory space. For brevity, the *delete* operator is left out of the discussion.

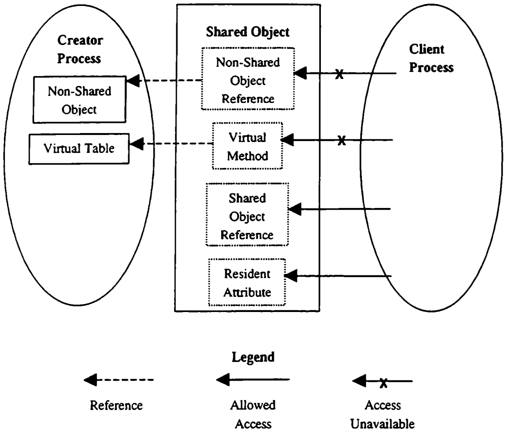


Figure 3 Access to Shared Object in Shared Memory

valid starting address. UNIX variants may reserve ranges of addresses for specific uses (with scant documentation on this topic). The address for the shared memory segment must not collide with these reserved spaces or addresses assigned to the program's text segment or stack; if it does, the operating system will reject the attach request. Sometimes, the developer must resort to compiling a version that lets the OS pick the starting address and prints this address. Then the developer can reconfigure the simulation to use this OS-picked address.

Shared objects that reference other shared objects are unavoidable in a flexible, configurable simulation. In fact, relying on object relationships is the most efficient means for processes to find shared objects. For example, a list of aircraft in the simulation is placed in shared memory. Processes use the list to find the location of each aircraft. The processes could then use the aircraft to find its landing gear model, etc. This method does not significantly differ from the way in which processes find their non-shared objects. The main difference is that the processes need a map to locate the first objects in these relationship chains. This map must exist at a fixed address in shared memory. These maps should contain pointers to very few items; the design should rely on object relationships as much as possible.

Shared objects can also reference non-shared objects. These references are only valid for the process that created the non-shared object. If another process attempts to access the non-shared object, a segmentation violation will result. Developers must ensure that all the objects they intend to share across processes are placed in shared memory.

However, there is one non-shared attribute that developers have no control over. Objects with virtual methods have a hidden pointer to a virtual method table. This prevents processes from calling polymorphic methods on shared objects, which the process did not create. C++ implements virtual methods using a virtual method table (vtable). The vtable contains a list of function pointers. Each entry in the list is associated with a virtual method. The pointer in the entry refers to the actual implementation that will be run when the virtual method is called on an instance of a particular class. When a class redefines a virtual method, the entry for that virtual method in the class's vtable is assigned a pointer to the class's implementation. Since objects of the same class would have identical vtables, most compilers attempt to create one vtable for all instances of the class. The vtable is a global object in the process's address space. The class obtains a hidden attribute, a pointer to its vtable. When a virtual method is invoked, the program references the vtable using the vtable pointer. Then, it looks up the appropriate method to call in the vtable and executes it. A shared object with virtual methods references a vtable in the process that created it. Other processes attempting to call a virtual method on this object will experience a segmentation violation.

Even if the developer finds a way to place the vtable in shared memory, most UNIX variants do not support the placement of functions in shared memory. Thus, the pointers in the vtable can only reference addresses in the creator's address space. The inability to invoke virtual methods on a shared object severely restricts the abstraction of shared objects. Clients must know the exact class from which they are requesting services. For example, a simulation may be required to support multiple atmospheric models. The developers initial inclination is to design a base class called Atmosphere with a virtual method called calculateProperties(). Each model is represented by a derived class that redefines the calculateProperties() method. However, all processes must share this class since the atmospheric properties are an input to all the vehicle models. Thus, the designer must bundle all of the atmosphere models into one class and provide non-virtual methods to execute each model.

The process that created the shared-object can still execute the object's virtual methods, however. In LaSRS++, each process keeps track of which vehicles it created. Those processes are responsible for calling the virtual methods, such as the doOperate() method introduced earlier.

All processes can still execute the non-virtual methods of a shared object. In this case, the separate address space presents an advantage. If two processes call the same method on two different objects, two different copies of the method are executed. There are no situations where the same copy of a method will be called concurrently. The developer does not need to be concerned with designing functions that are re-entrant.

<u>Multiple Threads</u>

Sharing objects in a multi-threaded environment is simple. All objects and all threads live in the same address space. Any thread has access to any object. The multiple thread design eliminates many of the restrictions found in the multiple process design. Developers do not need to create custom allocators. All object references are valid no matter which thread created the referenced object. Any thread can call a virtual method on a shared object that it did not create. In this sense, multi-threaded design is better suited to object-oriented programs.

However, the common address space does present a new challenge. Not only do multiple threads share data; they also share functions. When two threads invoke the same method on two instances of the same class, both are operating the same copy of the method. This situation is not safe unless the method was designed to be re-entrant.

Re-entrant methods do not intentionally make possible the concurrent modification of the same object. Since each thread receives its own stack, local variables and method arguments are guaranteed to be safe from concurrent modification. (The reference in a pass-by-reference argument is the data actually stored on the stack and is safe from concurrent modification. The referenced object is not protected.) All attempts should be made to localize variables used by the method. (Local variables that are declared *static* do not exist on the stack. They are a form of global data.) Re-entrant methods avoid direct modification of global or heap data (i.e., non-local data). (If non-local data that the method directly references is constant or guarded by a mutex, the method is essentially re-entrant.) Access to non-local data should be done through the method's argument list. If the argument list contains mutable pass-by-reference arguments, client code can pass the same object to multiple, concurrent invocations of a method, causing the same object to be modified. However, this does not invalidate a method as re-entrant. A re-entrant method only guarantees that the method itself will not modify the same object concurrently, outside of the control of the client. A re-entrant method does not prevent clients from causing concurrent modification of objects.

Good object oriented design leads to re-entrant methods. A well-designed object encapsulates its data, allowing access only through its functional interface[§§]. The methods in a class should only interact directly with attributes of the class or with local variables. The need to obtain information from outside of the class should be rare. When needed, external data should not

[§§] In C++, a class can expose its encapsulated data to another class or function through use of the *friend* keyword. Friendship breaks encapsulation and should not appear in objects intended for multi-threaded applications.

be obtained through global access. Outside information should passed through method arguments. If possible, arguments should be pass-by-value. Pass-by-value makes a local copy of the argument; modifying the copy does not modify the argument. However, pass-by-value is computationally efficient only for intrinsic types and very small objects. Larger objects are usually passed by reference. Pass-by-reference introduces an alias that could lead to concurrent data modification. If developer does not intend that the method modify the referenced data, the developer can enforce this design by declaring that the reference points to a constant object. The method cannot directly modify constant objects or call their mutator methods. The compiler will issue a syntax error for any attempt to modify a constant object. Classes that meet the above definition have a high level of cohesion.

C++ implements method access to class at attributes in a thread-safe manner. Methods access class attributes through a hidden argument, a pointer to the class. The pointer is named *this*. When a method is concurrently executed on different objects by different threads, each invocation uses a different *this* pointer from the thread's stack and thus modifies a different object. However, a method is no longer re-entrant if it modifies a static attribute that is not guarded by a mutex. Static attributes are a form of global data; the attribute is shared among all instances of a class. Classes should avoid mutable attributes that are static. All static data should be constant.

Sometimes there are compelling reasons to use global data. The developer can use object-oriented techniques to easily identify and guard this data. Any mutable global data should be placed in a class utility. Class utilities are classes where all of the data and methods are static. The data should be encapsulated and accessible only through methods. Methods can guard the global data with a mutex, relieving clients of this responsibility. Static methods must be invoked using a scope qualifier. The scope qualifier identifies that the method call may access global data, helping developers identify client methods that may not be re-entrant[¶].

Conclusions

The object-oriented features of C++ simplify the construction of a simulation that supports multiple, heterogeneous vehicles. Classes can aggregate data and constructors. With a single line of code, an object of a class can be created with a full, initialized copy of all its necessary data. Classes bind functions to data. This binding provides a single, consistent mechanism for calling actions on class attributes. The developer is freed from designing how functions will manipulate the data representing particular copies of a vehicle model. Through polymorphism, clients can manipulate a heterogeneous collection of objects as if they were objects of a common base class. Yet, the clients will invoke behavior specific to the actual class of the object. Polymorphism allows developers to declare a method in a base class that derived objects can redefine, essentially separating interface and behavior.

Applying these object-oriented techinques to a multi-CPU environment requires further consideration. The developer must be cautious not to modify the same data concurrently and to make sure that all processes view changes in data. The latter is accomplished by using the *volatile* qualifier for all data that can be read by a different process than the one that modifies it. There are many ways in which the former can occur. The developer can limit the possible situations by frequent use of the *const* qualifier on methods and pass-by-reference arguments. Since the compiler enforces the *const*, the incidents of accidental changes to data, concurrent or otherwise are reduced. Applying *const* also helps developers identify mutator methods and, thus, situations were unsafe concurrent execution of methods may exist. To increase the number of constant and re-entrant methods, developers should also move as much data as reasonable to pass-by-value arguments and local variables. Developers should avoid the use of

[¶] There are other situations where a scope qualifier is used with a method call. The point here is that static methods can only be called using a scope qualifier.

global data, particularly static class attributes and static local variables. All other global data should be placed in class utilities where access to them can be controlled, including the use of mutexes to guard access. The use of global data encapsulated in a class utility is also more easily identifiable.

In a multi-CPU environment, concurrent access of a mutator method with other methods can lead to erroneous behavior and program failures. Developers must identify and prevent these situations. Mutexes can guard against concurrent access of class methods. Buffers can defer object attribute changes; the changes then occur at once under controlled conditions. Dividing the event loops into blocks of activity can separate large-scale use of mutator methods by individual processes from concurrent use of constant methods.

Two multi-CPU environments are best suited for object-oriented programs: threads and multiple processes using shared memory. Threads are conceptually the simplest to design for. However, all code and data are shared in a multi-threaded environment. This increases the possibility of unsafe concurrent access to data. It also introduces the problem entering the same copy of the method code concurrently. Unless the method was designed to be re-entrant, this concurrent execution of the method could lead to concurrent modification of data, particularly global data. Using multiple processes with shared memory requires more work. Developers must create custom allocators that place and initialize objects in shared memory. Shared memory must be attached at the same address for each process in order to maintain the validity of references to shared objects. Shared objects can reference non-shared objects. These references are only valid in the process that created the non-shared object; attempts by another process to access them will lead to program failure. Because of the manner in which virtual methods are implemented, processes cannot execute virtual methods on shared objects that they did not create. Despite these restrictions, the multiple process design has some distinct advantages. The developer has control over which objects are shared. Each process has its own copy of class methods; this design does not have the same problem of method re-entrance that the multi-thread design has.

LaSRS++ exemplifies these techniques. It is a successful implementation of an object-oriented simulation that supports multiple, heterogeneous vehicles running over multiple CPUs.

Bibliography

[1] Booch, Grady. *Object-Oriented Analysis and Design With Applications.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994. ISBN 0-8053-5340-2.

[2] Stroustrop, Bjarne. *The C++ Programming Language.* Addison-Wesley Publishing Company, Reading, MA, 1997. ISBN 0-201-88954-4.

[3] *Working Paper for Draft Proposed International Standard for Information Systems - Programming Language C++.* ANSI X3J16/96-0225, 1996.

[4] Robbins, Kay A., Robbins, Steven. *Practical UNIX Programming: A Guide to Concurrency, Communication, and Multi-threading.* Prentice Hall, Upper Saddle River, NJ, 1996. ISBN 0-13-443706-3.

[5] Gallmeister, Bill O. *POSIX.4: Programming for the Real World.* O'Reilly & Associates, Inc., Sebastopol, CA, 1995. ISBN 1-56592-074-0.

[6] Meyers, Scott. *Effective C++: 50 Specific Ways to Improve Your Programs and Designs.* Addison-Wesley Publishing Company, Reading, MA, 1992. ISBN 0-201-56364-9.

[7] Quatrani, Terry. *Visual Modeling With Rational Rose and UML.* Addison-Wesley. Reading, MA, 1998. ISBN 0-201-31016-3.

AIAA-98-4531

A METHOD TO INTERFACE AUTO-GENERATED CODE INTO AN OBJECT-ORIENTED SIMULATION FRAMEWORK

Patricia C. Glaab, Kevin Cunningham[*], P. Sean Kenney,
Richard A. Leslie, David W. Geyer, Michael M. Madden[*]

Unisys Corporation
NASA Langley Research Center
Mail Stop 125B
Hampton, VA 23681

Abstract

Sophisticated computer-based tool packages allow control system researchers to develop and analyze complex control systems from their desktops. Tools such as Matrix X and Matlab are capable of producing auto-generated code of control system diagrams as an output option. Validation requirements are greatly reduced when the auto-generated code can be directly installed into an engineering simulation program.

The code produced by automatic code generators, however, is often cryptic and inflexible. Incorporating it elegantly into an object-oriented simulation requires forethought on the part of the simulation programmer. This paper presents a method of interfacing auto-generated code into a flexible, object-oriented simulation framework that maintains encapsulation of objects within the framework and does not compromise the design of the system.

Introduction

Support for flexible software design methods is a goal of object-oriented simulation program development at the NASA Langley Research Center (LaRC). The ability to accommodate auto-generated code from sophisticated computer-based tool packages provides a versatile analysis environment for researchers and potentially reduces the workload of the simulation programmer. The auto-generated code module, however, has very little flexibility when it must be ported directly from the tool output to the simulation environment. A technique was required to interface between auto-coded modules and the simulation framework that did not compromise the object-oriented design of the simulation program.

The methodology presented here was developed to support the baseline simulation of a commercial transport autopilot/flight director system (AFDS) developed within the Langley Standard Real-time Simulation in C++ (LaSRS++) [2] at LaRC. The AFDS implementation in the simulation was structured to allow auto-generated code to be easily incorporated into the C++ simulation program. Specific requirements for this design included two major objectives. The final design required rapid linking of auto-coded modules into a soft real-time simulation without having to modify simulation or auto-generated code. This was intended to minimize the potential of accidentally changing the code's behavior from its original configuration. Addi-

[*] Senior Member, AIAA

tionally, hand-coded versions of all AFDS subsystem objects were required as part of the baseline, and either of the two versions of each component must be selectable between runs. For the auto-coded versions, the intention was for a smooth and logical transition from a desktop tool environment to a soft real-time environment, and eventually to a hard real-time session with a cockpit and pilots. Other simulation users required the unmodified AFDS at a much earlier date for support of other studies, and since not all subsystem components would necessarily be delivered as auto-code, a complete hand-coded version was required as the default. The method developed allows the simulation user to select the version of the AFDS, baseline C++ or linked auto-code, for each component of the 757 autopilot independently.

The intention of this paper is to present a design method for managing complex, interacting systems with a sound overall object-oriented (OO) policy that ultimately allows flexibility at the lowest object levels. By designing a system architecture that meets these goals, incorporating code with special input/output (I/O) requirements is simplified at startup and for future maintenance. Auto-generated code created by computer-based engineering tool presents special I/O requirements because the tool manufacturer chooses the style of the interface and execution. These conventions must be gracefully accommodated if the code is to be ported to the simulation environment unaltered.

Like many legacy systems that were written before the advent of OO design philosophy, the prior implementation of the system relied heavily on component interaction and data sharing in a web-like fashion. The task of implementing an OO system heirarchy within the target system as a precursor to interfacing auto-code was included because this problem presents itself for many complex, legacy systems. It is also crucial to the clean incorporation of the auto-generated code. Whenever possible, established design patterns were used in the system architecture. Design patterns offer simple and elegant solutions to specific problems that tend to occur over and over in software problem solving. By incorporating time-tested solutions, a programmer may capitalize on the experience and efforts of many developers and begin a design at a stage that may have otherwise required years of refinement [1].

An Overview of the Example System

The documentation received for the simulated AFDS defines the system as a complex interaction of many components. Each of the components has unique I/O requirements, and all are highly dependent on each other. Though these component definitions provide a logical delineation for class objects, their interdependence violates encapsulation and would make unit testing of individual subsystem components difficult.

A hand-coded C++ implementation was required to function as part of the simulation framework to emulate standard behavior of the AFDS. This version was developed using the same style guide conventions imposed on the overall simulation program. The second version was an auto-generated code module delivered by researcher engineers using a computer-based engineering tool package. The behavior of this version would change as research required. The complexity of accommodating two different I/O format requirements for many components portended a maintenance nightmare. Also, the delivery of each auto-coded module was uncertain.. The absence of any or all auto-code modules could not break the system. Rapid reconfiguration and testing was required as the auto-coded components were delivered or modified. Three design criteria were established to meet final code requirements:

- The web-like interaction of the components as presented in the original documentation had to be removed and the I/O managed more cleanly
- An isolated location had to be created to perform the specific I/O requirements for the auto-code without propagating knowledge or burden of these special requirements to other parts of the system
- A foundation had to be laid to effectively isolate the internal complexity of component objects from the client code that would use them

Step One: Untangling the Web

An ideal unit test scenario allows the developer to begin testing at the lowest level in an isolated development area without having to depend on other objects being linked in or emulated. The object should then be portable to the production environment without modification. With a web of objects that directly interact with each other, unit testing becomes extremely difficult. References to data sources included in a production version of a class must be removed, or the referenced sources must be included in the test environment. As links within the class grow, so does the size and complexity of the unit test environment.

Though loose coupling of objects is common in modern OO code designs, isolation of code modules in legacy system may not be so clearly defined. The quick and easy inclusion of common blocks in FORTRAN coding, for example, is conducive to a final design that uses a web of data interaction.

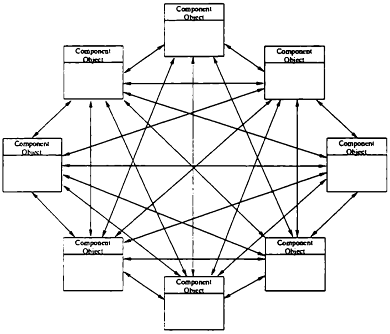


Figure 1 - Web of interaction when object are allowed to refer directly to each other

The design of the example AFDS system was developed in a procedural FORTRAN environment and relied heavily on information sharing and passing between objects. As a first step in untangling the interactive web, a Mediator class [1] was created to manage information to the subsystem components. The Mediator class encapsulates how the objects within the system may interact and orchestrates the exchange of inputs and outputs between subsystem components. This code pattern promotes loose coupling by keeping objects from referring to each other directly, and it lets the developer vary their behavior independently. By imposing this Mediator to control interaction, the tight coupling shown in the previous illustration was removed.

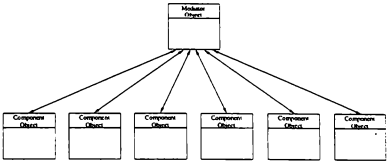


Figure 2 - Controlled interaction using a Mediator class

Each component object receives inputs from and provides outputs to the Mediator class exclusively. No component within the system has knowledge or dependence on any other component. This loosely coupled system removes interdependencies of subsystems and provides a clean, manageable point of reference for all subsystem I/O.

Step 2: Isolating and Adapting the Special I/O Requirements of the Auto-Generated Code

With auto-generated code delivered by an outside source, the simulation developer may not have control over the style of data input or function execution. Chances are slim that the manufacturer of the computer-based tool package chose the same style guide conventions adopted by the target simulation program. When several versions of a subsystem component are required, this necessitates several styles of communication to handle data transfer and operation. If the auto-code version is not stable (subject to frequent changes in a research environment, for example), the changing communication requirements translate to maintenance overhead that can propagate through the entire subsystem. This maintenance overhead can be isolated within one class with the use of an Adapter pattern [1].

The Adapter class converts the interface of a class into another interface that the client expects. This allows the client to communicate with the auto-code in a fashion consistent with local style guide conventions. A standard method of invoking execution of the auto-code can also be defined that need not be changed if the delivered auto-code changes significantly. The Adapter class can simply redefine the action upon the auto-code internal to itself.

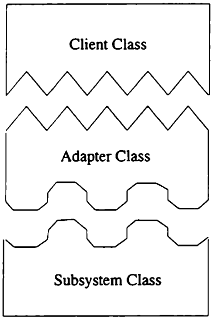


Figure 3 - Adapter class addition to translate interface

Adding the Adapter class layer becomes especially beneficial when the information sent to the various component versions is generally the same. In this case, the interface can be defined in one base class from which all components interfaces inherit. This isolated communication point provides an ideal place to set up a Facade.

Step 3: Hiding the Complexity of Component Behaviors and Interfaces

The decision requirements to accommodate two or more interface versions for each subclass component present a significant increase in the complexity of the client class (the Mediator class in the example system). Additionally, maintenance work may be required if interface requirements change. To remove knowledge of the complex interactions of the subsystems from the classes that use them, a unified interface can be defined to make the subsystem to make easier to use. This unified interface is called a Facade [1] and was used in the example system to encapsulate decision-making overhead within the operations of the subclass components themselves. In the C++ implementation of the AFDS component, the Facade is the base class from which the hand-coded implementation inherits. It is also the base class from which the C++ Adapter class to the auto-generated code inherits.

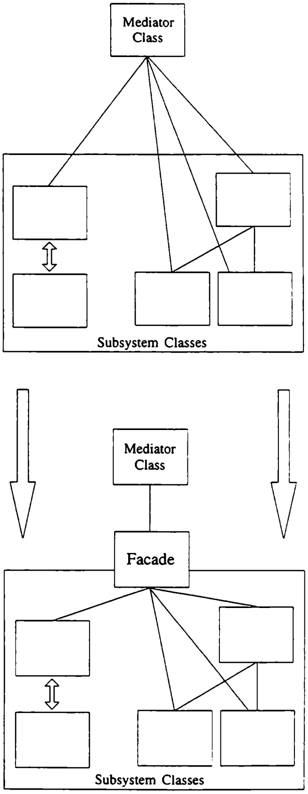


Figure 4 - Facade addition to reduce apparent complexity to client

In a simpler scenario where only one version of a subsystem must be interfaced, the Facade is not necessarily needed as part of the design. This is true as long as the client class, the Mediator, is only required to communicate to an interface isolated within one subsystem class and that interface is stable. For example, in this implementation with only auto-generated code as the subsystem target, the Mediator would speak directly to the Adapter class.

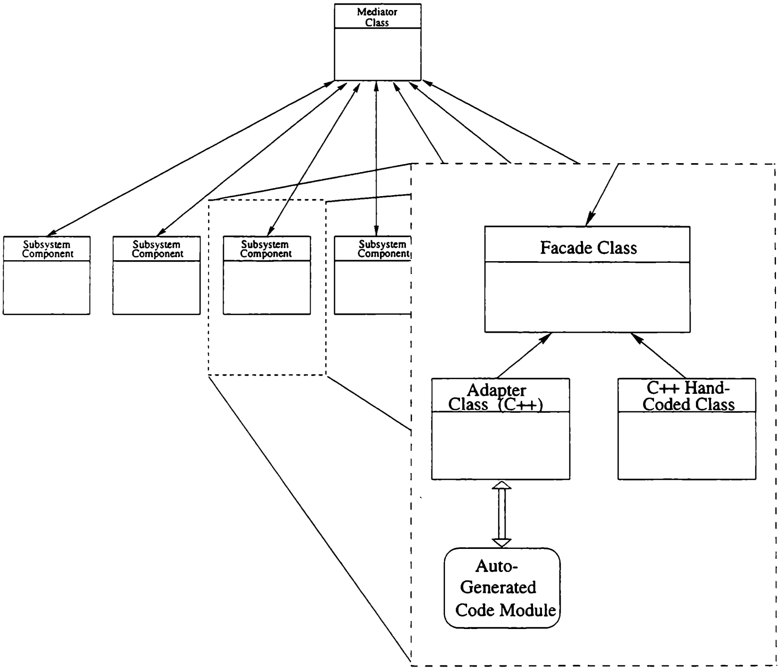
Unit Testing

Unit testing is handled easily at several levels in the final system architecture. The auto-generated version of the system is an isolated entity and its inputs and outputs and internal functionality are easily exercised in a stand-alone fashion. The hand-coded subsystem object uses an isolated, well-defined interface and a unit test program can either be imposed at the level of the Facade for exercising the client only, or above the Facade to exercise the Facade interface to the client code.

At a higher level, the two implementations can be tested or compared from a single unit test program which takes advantage of the one unified Facade interface.

Final Design and conclusions

When inflexible interfaces must be accommodated into an OO environment, a clean initial design of the system is critical to isolating the external component from the



simulation framework. Isolation of the interface then minimizes compromises that must be make to style guide conventions. By using the techniques outlined in this paper, an AFDS was implemented in the LaSRS++ framework that met the required design criteria. All code specialized to interface to auto-generated code modules is encapsulated into one class per component and hidden from the clients that use it. Code maintenance to support auto-code changes are localized within this one Adapter class. Selection of component versions is provided to the simulation user, yet facilitated to the information handling class (the Mediator) through the use of a Facade software pattern that provides one unified interface to all subsystems. The example system describes implementation of Xmath/SystemBuild auto_code in C, but theoretically the technique could be used to interface any type of code with special I/O requirements. A related approach is taken in the hardware interface of the LaSRS++ simulation with cockpit controls, displays, flight management computers, etc. as a method of isolating hardware-specific code requirements through abstraction [6].

Bibliography

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Massachusetts, 1995.

[2] Richard A. Leslie, et al. LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft. Paper Number AIAA-98-4529, August, 1998.

[3] Keith D. Hoffler, Dr. Thomas E. Alberts, and Patricia C. Glaab. Implementing the Control Laws for the NASA Transport Systems Research Vehicle B-757. ViGYAN, Inc., Hampton, Virginia, 1997.

[4] Steve McConnell. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, Redmond, Washington, 1993.

[5] Scott Meyers. Effective C++. Addison-Wesley, Reading, Massachusetts, second edition, 1998.

[6] P. Sean Kenney, et al. Using Abstraction To Isolate Hardware In An Object-Oriented Simulation. Paper Number AIAA-98-4533, August, 1998.

[7] Grady Booch. Object-Oriented Analysis and Design. Benjamin/Cummings, Redwood City, California, 1994.

MANAGING SHARED MEMORY SPACES IN AN OBJECT-ORIENTED REAL-TIME SIMULATION

David W. Geyer, Michael M. Madden*, Patricia C. Glaab,
Kevin Cunningham*, P. Sean Kenney, Richard A. Leslie

Unisys Corporation
NASA Langley Research Center
MS 169
Hampton, Virginia 23681

Abstract

Sharing memory spaces between parallel processes is a common practice in a real-time simulation environment. In an environment where parallel processes can exist on the same machine or on different machines, it is a challenge to develop and maintain reusable software that interfaces with shared memory spaces. The two main problems involved in dealing with shared memory spaces are: 1) providing platform independent access to the memory space, and 2) providing to all processes a consistent view of the structure and content of the memory space. Object-oriented techniques were used to create a software architecture designed to manage shared memory spaces. Object-oriented design patterns were used to present client code with a conceptual model of shared memory spaces while concealing the underlying implementation details. The resulting architecture is largely computing platform independent, with platform specific code being isolated to a few selected classes. The design was implemented in C++ for the NASA Langley Standard Real-Time Simulation (LaSRS++) Application Framework. This paper provides an overview of the design and implementation of the LaSRS++ shared memory management software.

Introduction

This paper describes a general object-oriented software architecture designed to manage shared memory spaces in a platform independent manner. This architecture focuses on the management of specific blocks of memory within a shared memory space. The software is designed to allow client code to query for existing memory blocks or to create new memory blocks. There is no provision (currently) to delete existing memory blocks. The architecture only provides the database functionality of keeping track of existing memory blocks and creating new memory blocks. It is the responsibility of other classes in the LaSRS++ framework to deal with how the memory blocks are actually used by client code.

Shared Memory Spaces

A process, or task, is the executing instance of a program. Each process is composed of an addressable memory space, a set of machine instructions to execute, and other entities maintained by the operating system. It is possible to map, or overlay, a portion of this memory space with the memory space from another process. Most modern operating systems provide mechanisms to perform this mapping with another process on the same machine. This mapping is known as shared memory and is part of the more general operating system facility known as Interprocess Communication. Both processes view the shared memory space as part of their own addressable memory space. As a result, any changes to the contents of the shared memory space by one process are

*Senior Member, AIAA

visible to the other process. This type of shared memory will be referred to as physical shared memory.

A similar type of shared memory space can be achieved between processes on different machines with the use of specialized hardware. One example of this type of specialized hardware uses a networking concept known as replicated shared memory.[1] Replicated shared memory uses a serial ring network with local memory modules at each network node. Device driver software allows a given process to map the local replicated memory for the respective node into the memory space of the process. This makes the replicated memory appear just like physical shared memory. When a process writes a value to a location in the replicated memory space, the network card reflects the value to the replicated memory space of all the other nodes in the ring. This is accomplished by transmitting both the value and the memory offset of the value across the network. As a result, the value is written to the same memory offset in all of the cards on the network.

It is also possible to pass interrupts between CPUs using a replicated shared memory network. A given CPU can transmit and/or receive interrupts based on changes made to shared memory locations. The same mechanism that is used to transmit data is also used to transmit interrupts to every CPU on the ring. It is up to the software running on each and every CPU to determine the effect of the interrupt. A CPU will only receive interrupts for shared memory locations for which the CPU actually chooses to receive interrupts. A CPU can ignore or handle interrupts based on its own criteria.

The replicated shared memory approach combines the benefits of physical shared memory with a fast message passing network. The result is fast internode communication speed with little software overhead. Bohman[1] provides a more detailed discussion of the benefits of combining physical shared memory with message passing.

SCRAMNet+ Network

The Shared Common Random Access Memory Network (SCRAMNet+), produced by Systran Corporation, is a commercial implementation of a truly general purpose replicated shared memory network. The nodes in a SCRAMNet+ ring can be connected via coaxial or fiber optic cable. The delays incurred when communicating between nodes is on the order of microseconds.

Systran provides a library of low-level C callable functions for the various operating systems that might be used on a SCRAMNet+ ring. The interface to these functions can be different, depending upon the platform that is being used. In order to write client code that is as platform independent as possible, it is necessary to normalize the interface to the SCRAMNet+ low-level functionality.

Platform Independent SCRAMNet+ Access

A SCRAMNet+ network allows a group of heterogeneous processors to function together as if part of a single multiprocessing machine. Each different machine on the SCRAMNet+ ring often uses its own operating system and a distinct version of the SCRAMNet+ device driver. Using object-oriented techniques, it is possible to present users on all platforms with a common low-level interface to SCRAMNet+ but allow the actual low-level implementation to vary according to the platform being used. This is accomplished by using several different object-oriented design patterns.[2] Design patterns describe simple and elegant solutions to specific problems in object-oriented software design.

Bridge Pattern

The Bridge pattern decouples an abstraction from its implementation. This design uses the Bridge pattern to isolate client code from the platform specific details of a specific implementation. The approach is to have clients use an abstraction object that forwards its public member function calls to a hidden platform specific implementation object. The abstraction object uses the implementation object through the pure polymorphic interface defined by the abstract implementation base class. In this case, a SCRAMNet+ interface object interacts with a platform specific SCRAMNet+ implementation object through a polymorphic interface. An appropriate concrete implementation class is defined for each platform being used on the SCRAMNet+ ring. The appropriate concrete implementation object for a given platform is selected at run-time.

Abstract Factory Pattern

The Abstract Factory pattern is used to create the correct instance of the platform specific SCRAMNet+ implementation object at run-time. This creational pattern provides an interface for creating families of related objects without specifying their concrete classes. In this case, the family of objects are the platform specific implementation objects. The abstract factory object contains knowledge of the specific platform that is being used. The constructor for the SCRAMNet+ interface object invokes the `makeScramnetImpl` member function of the abstract factory object. This member function uses knowledge of the specific platform to return the appropriate implementation object for the given platform. This specific implementation object is stored as a hidden attribute of the SCRAMNet+ interface object.

Singleton Pattern

The Singleton creational pattern is used whenever it is necessary to ensure a class only has one instance, and provide a global point of access to the single instance. The SCRAMNet+ network used for real-time simulation at NASA Langley is designed such that any given machine contains at most one SCRAMNet+ network card. For any process, there is a one-to-one correspondence between the SCRAMNet+ interface class and the actual SCRAMNet+ network card. Therefore, any given process on such a machine only needs a single instance of the SCRAMNet+ interface class. So, the SCRAMNet+ interface class is implemented as a singleton.

For simplicity, the Singleton pattern is also used for the abstract factory that creates the SCRAMNet+ implementation object. This abstract factory is really just a constructor-time detail of the SCRAMNet+ interface class. Using the Singleton pattern allows the SCRAMNet+ interface class constructor to access the abstract factory directly.

Figure 1 shows the class diagram for the SCRAMNet+ low-level interface software. Notes are included to point out the classes that utilize the various design patterns.

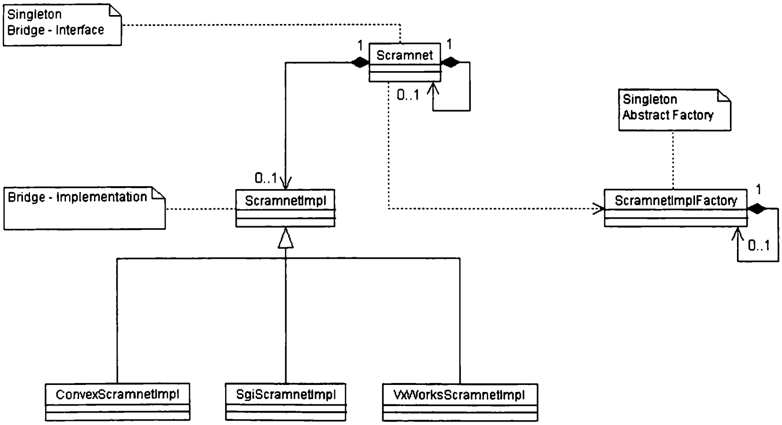


Figure 1: SCRAMNet+ Low-Level Interface Class Diagram

Design Advantages

This design has several advantages that address the issues of maintainability and portability:

1. *Client code is decoupled from the platform specific SCRAMNet+ implementation details.* This decoupling allows changes in the SCRAMNet+ implementation classes to have no impact on the client code; i.e., the client code does not need to be recompiled if the implementation changes. By coupling client code only to a common interface, the client code becomes platform independent. This allows client code to be portable to any platform supported by the SCRAMNet+ interface class.
2. *Only a single class needs to be written to support a new platform.* Adding support for a new platform involves two steps:
   (a) Write a new SCRAMNet+ implementation class that interfaces with the platform specific SCRAMNet+ device driver
   (b) Add to the SCRAMNet+ implementation abstract factory the capability to create the new SCRAMNet+ implementation object
3. *Creation of platform specific objects can be isolated to a single abstract factory object.* The SCRAMNet+ interface class only references the abstract SCRAMNet+ implementation base class. All concrete implementation details are confined to the implementation abstract factory.

Managing Shared Memory

The problem of managing a shared memory space involves providing all processes a consistent view of the structure and content of the memory space. It is assumed that a memory space is being shared between processes on the same or different machines. It is also assumed that the shared memory space is going to be subdivided into multiple separate memory blocks. Therefore, the processes are going to require:

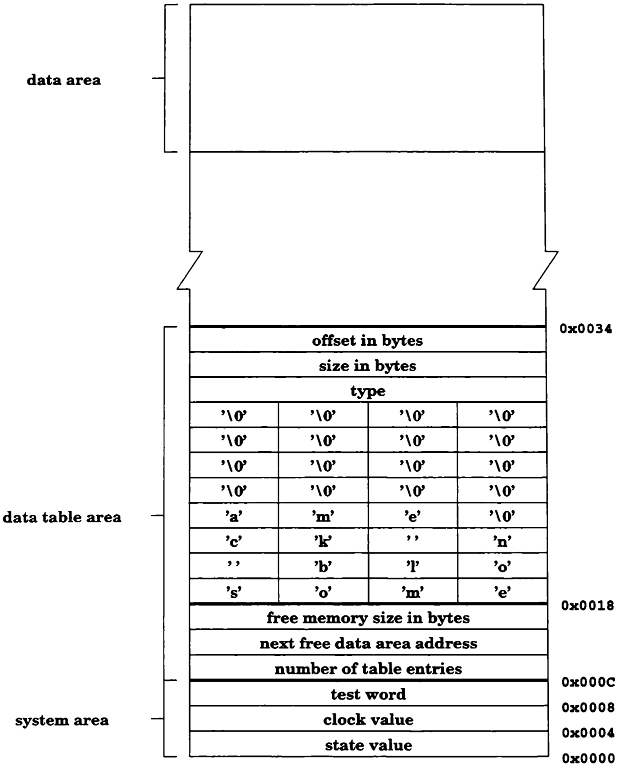
- the ability to acquire information concerning the memory blocks inside the memory space
- the ability to create new memory blocks inside the memory space

Design Strategy

The strategy used to satisfy these requirements involves using part of the shared memory space as a record keeping area to keep track of the allocated memory blocks. As the shared memory space is subdivided into smaller blocks, a record is generated for each memory block and stored in the record keeping area. Each record contains: block name, block type, block size in bytes, and block offset from starting address of the shared memory space. The record keeping area allows different processes access to information concerning the current usage of the shared memory space.

The shared memory space is divided into the following three areas:

1. System Area
   This area has a fixed size area and is used for data that pertains to all memory blocks.
2. Data Table Area
   This area is a data structure composed of bookkeeping values and a variable-length list of records that describe the arrangement of data blocks in the Data Area. Each record describes a separate, distinct memory block.
   Each record is composed of:
   - a fixed sized string for the block name
   - an integer to denote a user defined type for the memory block
   - the size of the block in bytes
   - the offset of the block from the beginning of the memory space
3. Data Area
   This area is a where the actual memory blocks are allocated.



offset in bytes			
size in bytes			
type			
’\0’	’\0’	’\0’	’\0’
’\0’	’\0’	’\0’	’\0’
’\0’	’\0’	’\0’	’\0’
’\0’	’\0’	’\0’	’\0’
’a’	’m’	’e’	’\0’
’c’	’k’	’ ’	’n’
’ ’	’b’	’l’	’o’
’s’	’o’	’m’	’e’
free memory size in bytes			
next free data area address			
number of table entries			
test word			
clock value			
state value			

Figure 2: Three area partitioning of shared memory space

Figure 2 shows the internal structure of a sample managed memory space. The base address of the memory space is at the bottom of the diagram and the hexadecimal numbers on the right side are offsets from the base address. The system area is always located at the beginning of the shared memory space. New data records are added following the system and bookkeeping areas toward the end of the memory space. New memory blocks are sequentially allocated from the end of the memory space toward the data records. This arrangement gives the memory block management scheme maximum flexibility in allocating a variable number of variable length memory blocks.

High Level Design

The three area strategy described above is encapsulated in the `MemoryBlockManager` class. A `MemoryBlockManager` object applies the strategy to any raw memory space with a known base address and size in bytes. The `MemoryBlock` class is an abstraction of an allocated block of memory. Every memory block has an associated name, a base address and a size in bytes. The interface of the MemoryBlockManager class uses MemoryBlock objects to convey information about allocated blocks to client code.

The MemoryBlockManager class responds to successful memory block requests by returning one or more MemoryBlock objects, depending on the request criteria. The request criteria can be:

- memory block name
- all memory blocks of a specified type
- all memory blocks associated with a given machine name
- all memory blocks of a specified type associated with a given machine name
- all memory blocks in the memory space

A single MemoryBlock object is returned by a successful create memory block call. Client code provides the new memory block name, type and size in bytes.

Managing SCRAMNet+ Memory

Managing a SCRAMNet+ memory space is now simply a matter of combining the SCRAMNet+ interface object with an instance of a MemoryBlockManager object. The ScramnetMemoryBlock class (derived from the MemoryBlock class) is also available that combines the notion of a memory block with the ability to interrupt enable/disable SCRAMNet+ memory locations. This abstraction is useful in decoupling client code from the SCRAMNet+ interface singleton. In practice, client code usually requests a ScramnetMemoryBlock object and selects memory locations inside the memory block to either send or receive interrupts. The client code does not need access to the majority of the SCRAMNet+ interface. The ScramnetMemoryBlock class combines this functionality into a single, easy to use class.

There are situations where it is necessary to prohibit memory block creation for certain processes on a SCRAMNet+ ring. This can occur for processes that are responsible only for monitoring or recording SCRAMNet+ activity. It can also occur in cases where centralized control of memory block creation is needed. The creation of memory blocks may be part of the overall system initialization process. In order to facilitate this level of restricted access, two separate, but related, classes are available.

The ScramnetReader class provides the following functionality:

- uses the SCRAMNet+ interface object to apply a MemoryBlockManager object to the SCRAMNet+ memory space
- forwards existing memory block requests to the same MemoryBlockManager object
- excludes the creation of memory blocks from its own interface to ensure access only to existing memory blocks
- provides an enumerated list of types for the memory blocks in the SCRAMNet+ memory space (These enumerators are used when requesting a memory block by type or when creating a memory block)

The ScramnetReader class uses the Singleton pattern in order to maintain a one-to-one correspondence with the SCRAMNet+ interface object and hence the single SCRAMNet+ network card in the machine.

The ScramnetManager class uses the ScramnetReader singleton to provide access to existing memory blocks. ScramnetManager also provides a create memory block request facility for client code by forwarding such requests to the MemoryBlockManager object used by ScramnetReader. The ScramnetManager class uses the Singleton pattern for the same reason as the ScramnetReader class. Both ScramnetReader and ScramnetManager return ScramnetMemoryBlock (i.e., specialized MemoryBlock) objects in response to request/create memory block calls.

Figure 3 presents a detailed class diagram for the SCRAMNet+ Memory Management software.

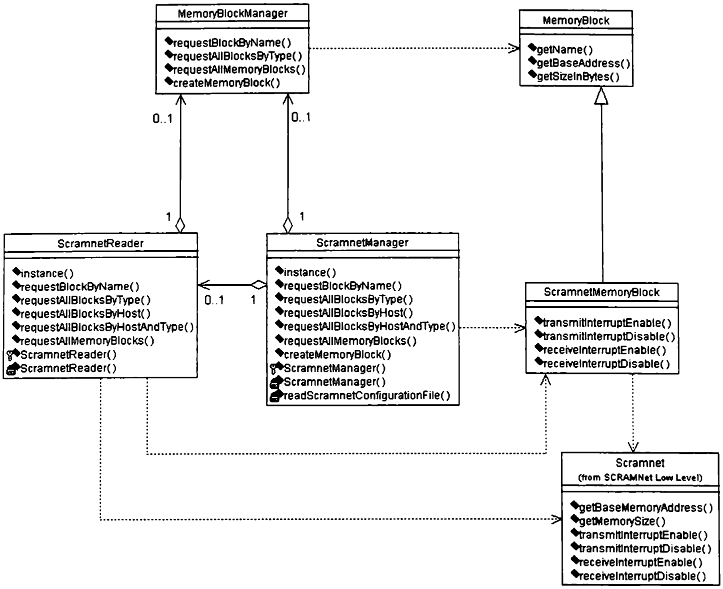


Figure 3: SCRAMNet+ Memory Management Class Diagram

Implementation Issues

Figure 4 presents a scenario diagram illustrating how the ScramnetReader singleton forwards an existing memory block request to the more generic MemoryBlockManager object. In similar fashion, Figure 5 is a scenario diagram illustrating how ScramnerManager singleton forwards a new memory block creation request to the same MemoryBlockManager object.

SCRAMNet+ objects are passive, so there is no direct communication between SCRAMNet+ objects executing on different processors. There is no notion of mutual exclusion between different nodes on a SCRAMNet+ ring. This means that a given node can not acquire exclusive access to a SCRAMNet+ memory location. There is no guarantee that a value written to a SCRAMNet+ memory location will not be overwritten by some other node on the ring. This issue is not dealt with by the SCRAMNet+ memory management software. Another layer of software is required that controls when the memory management objects are constructed and used. The system area at the start of the SCRAMNet+ memory space can be used transmit communication protocol information between SCRAMNet+ nodes.

One way to handle this issue is to have only one node in the ring be responsible for creating new SCRAMNet+ memory blocks via ScramnetManager. All of the other nodes on the ring access existing memory blocks using ScramnetReader.

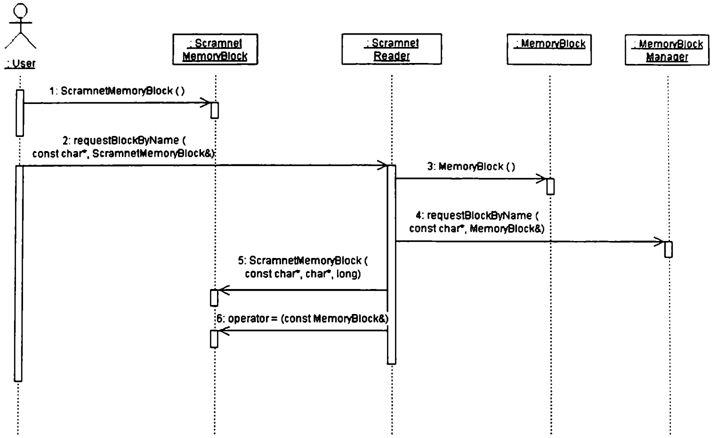

: User
: Scramnet MemoryBlock
: Scramnet Reader
: MemoryBlock
: MemoryBlock Manager
1: ScramnetMemoryBlock ( )
2: requestBlockByName (const char*, ScramnetMemoryBlock&)
3: MemoryBlock ( )
4: requestBlockByName (const char*, MemoryBlock&)
5: ScramnetMemoryBlock (const char*, char*, long)
6: operator= (const MemoryBlock&)

Figure 4: Requesting an existing SCRAMNet+ memory block by name

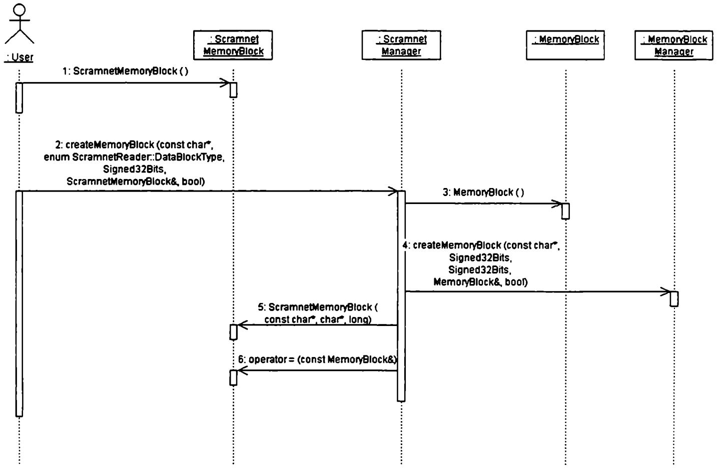

: User
: Scramnet MemoryBlock
: Scramnet Manager
: MemoryBlock
: MemoryBlock Manager
1: ScramnetMemoryBlock ( )
2: createMemoryBlock (const char*, enum ScramnetReader::DataBlockType, Signed32Bits, ScramnetMemoryBlock&, bool)
3: MemoryBlock ( )
4: createMemoryBlock (const char*, Signed32Bits, Signed32Bits, MemoryBlock&, bool)
5: ScramnetMemoryBlock (const char*, char*, long)
6: operator= (const MemoryBlock&)

Figure 5: Creating a new SCRAMNet+ memory block

Conclusions

A general-purpose method for managing shared memory spaces has been developed in an object-oriented environment. This management scheme is designed to be platform independent and has been utilized on a variety of different computing systems. The design of the software is constructed using well known object-oriented design patterns.

An object-oriented design is presented that normalizes access to SCRAMNet+ hardware across a group of heterogeneous computing platforms. Platform specific SCRAMNet+ code is isolated from the SCRAMNet+ interface code. This allows for greater reuse of client code that uses SCRAMNet+ since the client code is not directly dependent on the platform specific SCRAMNet+ code.

This general shared memory management design and the SCRAMNet+ low-level interface design are combined as the basis of a specific design to manage SCRAMNet+ memory spaces. The specific design uses a combination of composition and inheritance to achieve the desired functionality.

Bibliography

[1] T. Bohman. Shared-memory computing architectures for real-time simulation - simplicity and elegance. Technical Report D-T-SP-TPAIAA01-A-0-A1, Systran Corporation, February 1996.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

[3] R. Martin. *Designing Object-Oriented C++ Applications Using the Booch Method*. Prentice-Hall, Inc., 1995. ISBN 0-13-203837-4.

[4] S. Meyers. *Effective C++*. Addison-Wesley Publishing Company, 1992. ISBN 0-201-92488-9.

[5] T Quatrani. *Visual Modeling With Rational Rose and UML*. Addison-Wesley Publishing Company, 1998. ISBN 0-201-31016-3.

[6] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley Publishing Company, third edition, 1997. ISBN 0-201-88954-4.

[7] Systran Corporation. *SCRAMNet Network Programmer's Reference Guide*, June 1996. Document No. C-T-MR-PROGREF#-A-0-A4.

AIAA-98-4533

USING ABSTRATION TO ISOLATE HARDWARE IN AN OBJECT-ORIENTED SIMULATION

P. Sean Kenney, Richard A. Leslie, David W. Geyer,
Michael M. Madden*, Patricia C. Glaab, Kevin Cunningham*

Unisys Corporation
NASA Langley Research Center
Mail Stop 169
Hampton, VA 23681

Abstract

A common problem faced in the design of an object-oriented simulation is how a complex simulation model should interface with the simulator hardware. This paper describes a design that isolates the hardware interface from the complex models of a simulation environment. A detailed description of the design is provided and the advantages and disadvantages of the design are discussed. A working example of the abstraction as implemented in the Langley Standard Real-Time Simulation in C++ (LaSRS++) framework is also presented. Conclusions drawn from the experience of implementing the design are also given.

Introduction

Interfacing a complex simulation model with the associated simulator hardware is a common problem faced in the design of an object-oriented simulation. The resulting design should result in an interface that decouples the model from the simulator hardware in the system. A tightly coupled design unnecessarily complicates a system because a class becomes harder to comprehend, modify, or debug by itself.[1]

A desirable object-oriented design couples classes with commonality through inheritance while decoupling unrelated classes. Inheritance is one method to promote code reuse through common interfaces. A good design is also complete. This means that the interface of the class encapsulates all of the meaningful behaviors of the class. Finally, a desirable design creates classes that are unit testable. This allows the users to easily verify that a class is functioning properly and to easily test modifications.

The hardware abstraction presented in this paper is the product of an iterative object-oriented design process. The design provides a decoupled, unit-testable, and complete interface to a simulation framework. The abstraction is intended to simplify the complexity of connecting simulation models to simulator hardware devices.

The Hardware Abstraction

The hardware abstraction is composed of three main components: drivers, interfaces, and builders. The drivers are the classes that actually transmit and receive data with the simulator hardware devices, the interfaces are communication classes that pass data between a simulation model and a driver, and the builders construct all of the appropriate drivers and interfaces as needed. Figure 1 uses the Unified Modeling Language (UML) to demonstrate the relationships between the drivers, the interfaces, and the simulation models.

* Senior Member, AIAA

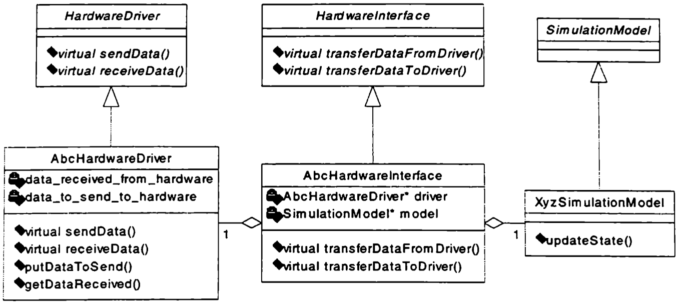


Figure 1 - Drivers, Interfaces, and Simulation Models

Drivers

The driver classes are the classes that actually transmit and receive data with the simulator hardware devices. They typically contain buffers to hold the data that is transferred with the hardware and member functions to access or modify the data buffers. In the above diagram, the class AbcHardwareDriver is shown to have defined the two virtual functions found in the abstract class HardwareDriver that send and receive data. The class also defines methods that access and modify data transferred to and from the hardware. The driver class therefore provides the abstract interface of HardwareDriver and an interface specific to the "Abc" hardware. The driver classes are an implementation of the Bridge design pattern.

The Bridge pattern decouples an abstraction from its implementation.^{2} This pattern is used whenever the implementation is to remain hidden from a client[†] and the particular implementation is selected at run-time. The pattern also allows the abstractions and the implementations to be extended through subclassing.

Interfaces

Interfaces are communication classes that pass data between the driver and the simulation models. The interface class also performs any manipulation of the data before transferring the data to its destination. The interface class is essentially a one way or two way data pump between the driver and the simulation model. In the illustration above, the AbcHardwareInterface is given a reference[‡] to the AbcHardwareDriver and the XyzSimulationModel when it is instantiated. The class would then use the two references to transfer data between the two classes when appropriate. The interfaces are a variation of the Mediator design pattern.

The Mediator design pattern keeps classes from referring to each other explicitly and encapsulates how the set of classes interact.^{2} The strongest asset of the Mediator design pattern is that it completely decouples the two classes from each other. It should be used whenever two classes are unrelated but need to communicate with each other.

Builders

Builder classes construct all of the appropriate drivers as requested by the user and construct all of the corresponding interfaces required by the simulation models. The builder classes provide an interface for creating the driver and interface classes without other

[†] Any object or function that operates on an object is a *client* of the object.

[‡] Unless stated otherwise, reference refers to both the reference and pointer types in C++.

simulation classes having knowledge of the particular concrete classes. The builder classes could be implemented as an Abstract Factory design pattern, a Factory Method design pattern, or any other creational design pattern.

The Abstract Factory design pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes. The Factory Method design pattern defines an interface for creating an object where the subclasses decide which class to instantiate.[2] Many other creational design patterns exist. The most appropriate pattern should be used for a particular application.

Design Advantages

The advantages of this design are:

1. *The simulation models are completely decoupled from the hardware driver classes.* This allows the models to be tested with or without simulator hardware. The behavior of the simulation models due to different inputs can be fully tested offline allowing comprehensive analysis of performance without using valuable simulator hardware resources. Once the performance of a model has been validated, the hardware inputs/outputs can be used to validate the model's performance in the simulation. This minimizes the validation required of new models. The decoupled simulation models also remain portable. A class hierarchy with an abstract interface defined by the base class allows different computation models to be incorporated into the simulation and use the existing hardware interface class without modification. The model classes may be exported to other sites without requiring any modifications for use. A model imported from another site can be "wrapped" in a class that has the interface required by the existing hardware interface, thereby quickly assimilating the new computational model into the simulation.

2. *Modifications to simulator hardware only require a change to the driver class.* Many hardware devices receive major and minor modifications over their lifetimes. Minor modifications are often changes to software, buffer sizes, etc. and require little or no change to the software used to communicate with the device. Major modifications may require a significant change to the software used to communicate with the device however. Because the driver encapsulates all of the code involved with communicating to the device, the simulation is completely isolated from the modifications.

3. *The hardware driver classes can be unit tested.* Any modifications to a driver class can be tested without the simulation model. A diagnostic program can be written that uses the driver to communicate with the hardware. The diagnostic program serves two functions. It can be used to test any changes made to the driver program and it can be used to verify the operation of the hardware prior to use by the simulation. Software configuration management eases the burden of testing the new driver class by allowing the user to verify that the hardware is operating correctly with a previous version of the driver before testing the new version. (This is usually not possible when the hardware has been modified).

4. *The driver and/or interface class may be used to emulate the hardware.* Often a simulation uses real-world hardware like a flight management computer to assist in research or testing. A software emulation of a hardware device can be placed in either the driver or interface class to allow the simulation to perform necessary communications with the emulated hardware when the real hardware is unavailable. The hardware emulation may also be modified to conduct research experiments.

5. *Changes to the models can not affect the communication between a driver and it's respective hardware.* A modification to a simulation model may result in bad data being transmitted to a hardware device, but it can not cause a connection loss or crash if the hardware interface class properly limits the data being sent to the hardware device.

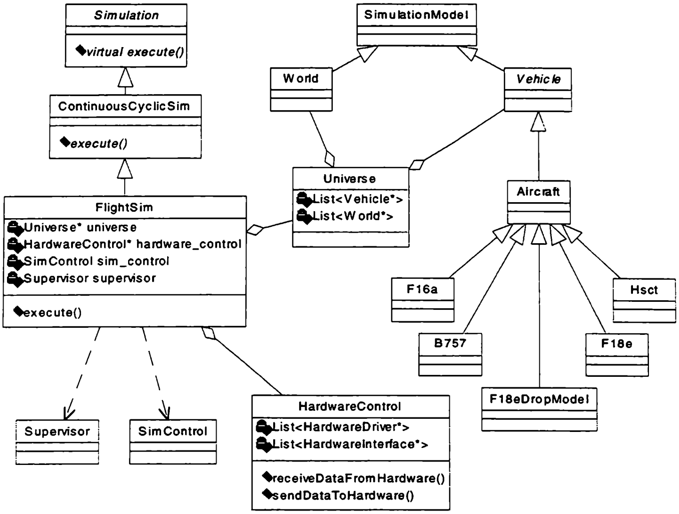


Figure 2 - LaSRS++ Framework

6. *The hardware interface classes are generally very trivial.* The classes simply use the accessor and modifier methods of the driver class and the simulation model to transmit data. Any calculations required when manipulating the data can easily be verified through testing.

7. *The hardware interface classes can often be reused by different simulation models without modification.* If the models share a common base class and the hardware interface only uses a reference to the base simulation model then no modification is required for use with different simulation models.

Design Disadvantages

The disadvantages of this design are:

1. *A modification to the interface of a simulation model requires the hardware interface class to accommodate the change.* Changing the interface of a class will always require the modification of any other classes that uses the aforementioned class. This is expected to present problems and cannot be avoided in any object-oriented design.

2. *A change to a hardware driver interface will require the appropriate hardware interface classes to be modified.* Again, this is the type of problem that is hard to avoid in a good object-oriented design.

3. *Public methods are required in model classes for all data needed by a hardware interface.* This requirement may force the designer of a class to add additional member functions to the class solely because a hardware interface needs the data. Usually a data item that is an output of a model already has public member functions to allow the class to be unit tested, so this require-

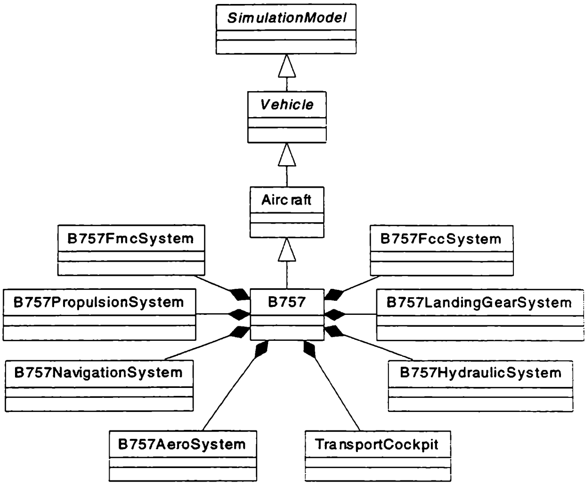


Figure 3 - Typical LaSRS++ Aircraft Hierarchy

ment is rarely a problem in a good object-oriented design.

4. *There is an added overhead associated with the interface class.* While the design decouples the driver classes from simulation models, it burdens the simulation with additional compute time to transfer the data between the two systems. Compilers provide the ability to minimize this overhead by inlining member functions.

The LaSRS++ Framework

The hardware abstraction presented in this paper is implemented in the Langley Standard Real-Time Simulation in C++ (LaSRS++) framework. LaSRS++ provides a powerful object-oriented framework for dynamic vehicle simulations in real-time, flight simulations in particular. The framework's object-oriented design makes the software extremely flexible, easily maintainable, and provides a high degree of reuse.[3] The framework is also portable and allows the user to run in either a hard or soft real-time environment.

LaSRS++ was designed to provide an efficient means to simulate dynamic vehicles of any level of fidelity. The framework allows n vehicles to be simulated on m CPUs and each vehicle may or may not be connected to a simulator hardware device.

Figure 2 illustrates several of the key components of the LaSRS++ framework. The framework supports real-time flight simulation through a class called FlightSim. This class manages the main event loop and instructs several objects to perform certain operations at specific times. FlightSim has references to HardwareControl and Universe and uses the Singleton classes SimControl and Supervisor. The Singleton creational pattern ensures that only once instance of class exists and provides a global point of access to this instance.[2] SimControl contains information about the simulation such as the current mode, the time

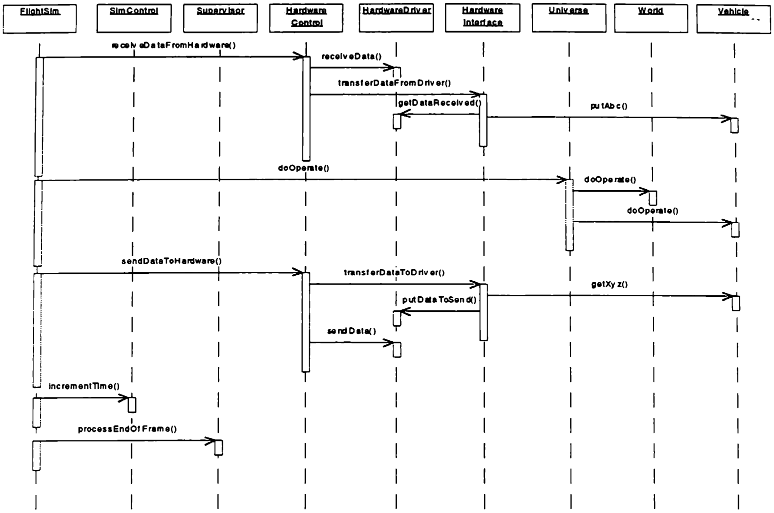


Figure 4 - Object Interaction Diagram for FlightSim

step, the current time, etc... Supervisor enforces that the simulation adheres to hard real-time when applicable. HardwareControl contains lists of all of the HardwareDrivers and HardwareInterfaces created for a particular simulation. Universe contains a list of Worlds and a list of Vehicles.

Figure 3 illustrates a typical aircraft hierarchy. The aircraft is composed of the different systems that make up the aircraft model. The illustration shows the B757 having systems for aerodynamics, propulsion, navigation, hydraulics, the landing gear, the flight management computer, the flight control computer and the cockpit. Any one of these systems may be used by a hardware interface class to obtain data from or provide data for a simulator hardware device. A hardware interface class may be passed references to the necessary aircraft systems when it is instantiated.

The simulation works in the following manner. First the drivers are constructed from scheduled hardware information by the hardware driver builder. The drivers are then placed on the hardware driver list. Next, the user selects what kind of vehicle to create, what the initial conditions of the vehicle are, which hardware the vehicle should establish a connection with, where the vehicle is located, etc... The new vehicle is then created and placed on the vehicle list. The vehicle's hardware interface builder now creates the appropriate hardware interfaces as selected by the user for the new vehicle and places them in the hardware interface list. Each hardware interface is passed a reference to the required simulation model and the appropriate hardware driver as it is instantiated. Additional vehicles and their hardware interfaces may also be created by the user in the same manner.

Figure 4 is an object interaction diagram for FlightSim and the key components of the simulation that it

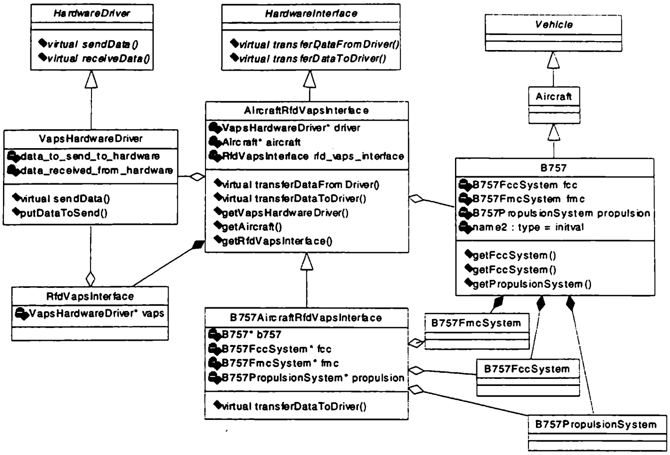


Figure 5 – A Typical LaSRS++ Hardware Interface

communicates with. At the beginning of any frame the first task of the simulation application is to march through the list of drivers and instruct the drivers to receive new data from the hardware. Next, the simulation steps through the interface list and instructs each interface to transfer data from the driver to the simulation models. The vehicles are then told to perform whatever computations they need to perform. Next, the hardware interfaces are told to access the new simulation model data and transfer it to the hardware drivers. Finally, the drivers send the data to the hardware devices. The process repeats until the simulation session is terminated.

The LaSRS++ Implementation

Recall that figure 3 illustrates the HardwareInterface classes exchanging data only with the Vehicle class. It should be noted that the hardware interface classes are capable of transferring data with not only the Vehicle classes but also with derived classes like Aircraft and B757 and any objects that these classes might contain if public accessor methods are made available. Figure 4 illustrates several of the component classes that make the B757 aircraft model. A hardware interface class specific to B757 can contain references to any of the components of B757 that it has access to. This allows the interface to directly access or modify the component system directly.

A typical LaSRS++ hardware interface is shown in Figure 5. The diagram illustrates the components used to communicate with the heads-down displays in the Research Flight Deck (RFD). The RFD is an aircraft cockpit at LaRC designed to facilitate B757 research projects. The heads-down displays used in the RFD were written with a program called VAPS (Visual APplicationS builder). The VAPS programs receive data to update the displays through a shared memory segment that is broken into channels. The VapsHardwareDriver class is responsible for allocating the shared memory segment needed to provide the VAPS program with new data.

The RfdVapsInterface class provides the means to separate the shared memory segment into the channels that the RFD heads-down displays require and the class also provides modifier functions to update the data in the channels. AircraftRfdVapsInterface class actually contains an RfdVapsInterface object along with a reference to the Aircraft for which it was created. The transferDataToDriver method uses the Aircraft reference to obtain all of the data required by RfdVapsInterface that is common to the Aircraft class. Angle of attack, mach, roll angle, pitch angle, and climb rate are examples of the data obtained from the Aircraft reference. The AircraftRfdVapsInterface also provides methods to allow the RfdVapsInterface object to be manipulated by a child class like B757RfdVapsInterface.

B757RfdVapsInterface is the B757 specific interface for the RfdVapsInterface. The transferDataToDriver method first calls the same method found in it's base class and then obtains data from B757 component systems to update any parameters in RfdVapsInterface that were not updated in AircraftRfdVapsInterface.

The design allows any aircraft to be used in a simulation in the RFD cockpit with the default heads-down displays without any modification. Because AircraftRfdVapsInterface uses a reference to Aircraft to obtain data to update the displays, the class may be used by any dynamic vehicle that inherits from Aircraft. This feature increases code reuse in the simulation framework.

The design also allows a child class to override the default behavior found in the parent class. B757VapsInterface can modify any of the data sent to the heads-down displays by AircraftRfdVapsInterface simply by resetting the data member to the desired value. This allows different units or values computed differently to be sent to the same display.

The hardware abstraction allows testing without the hardware in either hard or soft real-time (batch). Batch provides a means to test and debug modifications without tying up hardware resources. As mentioned above, the appropriate drivers are constructed from scheduling information. This allows the user to select which hardware interfaces are constructed at run time. In batch, hardware interfaces and drivers are exercised but in most cases no data is actually transferred with a hardware device because the drivers do not attempt to communicate to the hardware devices. Some of the drivers are configured to communicate with a hardware device using internet sockets to allow testing in batch. This reduces some of the hardware resource requirements while allow the simulation to communicate with a selected hardware device. In hard real-time, data transfers can also be turned off so that the hardware interfaces and drivers are exercised but no data is actually transferred with the hardware.

Cockpits

The cockpit hardware interfaces are treated slightly differently in the LaSRS++ framework. The interface class for many simulator hardware devices, out the window visuals for example, are not appropriate for containment by a vehicle in an object-oriented design. On the other hand, an aircraft certainly "has a" cockpit. To address this problem, the LaSRS++ framework provides each aircraft with a cockpit interface and allows any aircraft to run from any cockpit without the vehicle being aware of which cockpit it is connected to. An aircraft is given a cockpit interface when it is created. Currently an aircraft can have a TransportCockpit, a FighterCockpit, or a DropModelCockpit. A specific hardware interface for each cockpit interface determines how a specific cockpit behaves for a cockpit interface.

This design allows any aircraft to connect to any cockpit once the specific hardware interface for the aircraft's cockpit interface has been created. Duplication of code is avoided by placing common code in generic classes and having the specific hardware interface contain the generic class internal to themselves.

Conclusions

Because the hardware interface is abstracted away in a manner that keeps the actual simulator hardware communications hidden from the rest of the framework, the framework is a robust, portable, and easy to maintain simulation system. Continual reuse of the hardware abstraction ensures that new hardware interfaces can be easily added into the framework and that these interfaces can be easily tested and debugged with or without the hardware. The advantages of the design far outweigh the few disadvantages presented here. Although the abstraction was originally designed to support the NASA Langley Research Center flight simulation hardware, the design could be used in any object-oriented framework to heighten reuse and maintainability.

Bibliography

[1] Grady Booch. *Object-Oriented Analysis and Design*. Benjamin/Cummings, Redwood City, California,1994.

[2] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.

[2] Bruce Eckel. *Thinking in C++*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[3] Richard A. Leslie, et al. *LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft*. Paper Number AIAA-98-4529, August, 1998.

[4] Michael Madden, et al. *Constructing a Multiple-Vehicle, Multiple-CPU Using Object-Oriented C++*. Paper Number AIAA-98-4530, August, 1998.

[5] Patricia Glaab, et al. *A Method to Interface Auto-Generated Code into an Object-Oriented Simulation*, Paper Number AIAA-98-4531, August, 1998.

[6] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, Massachusetts, third edition, 1997.

[7] John Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, Reading, Massachusetts, 1996.

[8] Robert C. Martin. *Designing Object-Oriented C++ Applications Using The Booch Method*. Prentice-Hall, Englewood Cliffs, 1995.

[9] Scott Meyers. *Effective C++*. Addison-Wesley, Reading, Massachusetts, 1992.

[11] Scott Meyers. *More Effective C++*. Addison-Wesley, Reading, Massachusetts, 1996.

[12] David R. Musser, Atul Saini. *STL Tutorial and Reference Guide*. Addison-Wesley, Reading, Massachusetts, 1996.

[13] Terry Quatrani, *Visual Modeling With Rational Rose and UML*, Addison Wesley, Reading, Massachusetts, 1998.

AIAA-98-4534

APPLIED LOW COST "REAL TIME" FLIGHT SIMULATION CODE GENERATION IN AN AERONAUTICAL ENGINEERING CURRICULUM

Douglas K. Hiranaka, Graduate Student
Eric T. Vinande, Undergraduate Student
Douglas C. Cameron, Graduate Student
Dr. Daniel J. Biezad, Professor, AIAA Associate Fellow

Aeronautical Engineering Department
California State Polytechnic State University
San Luis Obispo, CA 93407

Abstract

The synergistic learning potential of hands-on experience with flight-controls simulation complementing the classical and modern classroom theories is now affordable for resource limited universities. Aeronautical engineering programs now have low-cost and powerful computing capabilities that provide the opportunity for a thorough exploration of flight simulation with flight control laws by real-time simulation on the desktop Personal Computer (PC) or on an easily reconfigurable fixed-based simulator driven by PCs. The student is now capable of quickly building the simulation model of a bare-airframe, designing the control laws to tailor the response types and flying qualities, then flight testing the resulting augmented aircraft via joy-stick inceptor on the desktop PC or electronic force feedback inceptor in a fixed-based simulator. Using Matlab/Simulink on a PC, consistent and verifiable real-time simulation code is auto-coded from block diagrams representing the direction of the simulation and architecture of the control laws. Work is in progress to simplify and facilitate the connectivity between hardware, such as inceptors and displays, to the real-time simulation.

Introduction

The development of Fly-By-Wire (FBW) flight control systems produces dramatic advances in aircraft handling qualities and performance. However, this increase in design flexibility requires additional training for the engineer in order to analyze these complex systems rapidly and cost-effectively. This training is based upon fundamental understanding of complex mechanical systems highly coupled with the fundamentals in FBW flight control systems. This large multitude of disciplines must be tied together with hands-on experience provided by pilot-in-the-loop simulations. Fortunately, affordable and flexible simulation capability is now available on a desktop PC or fixed-based simulator driven by PCs, like Cal Poly's PHEAGLE (Figure 1).

Until recently, demonstrating to engineering students the resulting handling qualities of their student designed FBW flight control systems is nearly impossible due to the high cost and large time required for engineers to generate the source

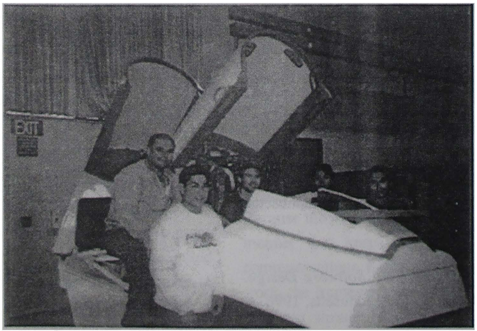


Figure 1. Cal Poly's PHEAGLE (a fixed base simulator)

code for the simulation and the control laws. The availability of low-cost and powerful PCs allow students to use the same tools that NASA and industry use to analyze and design control systems. For example, aerospace companies use Matlab/Simulink to analyze flight control systems[1,2]. Mathworks recently released the same tools for desktop PCs that have been used on workstations to conduct frequency and time domain analysis on complex flight control systems. Unfortunately, the frequency domain analysis provides little intuitive basis for the student. The hands-on learning that spawns mental connections between modal analysis and the time domain is best demonstrated by pilot-in-the-loop simulation. Fortunately, Mathworks provides the Real Time Workshop (RTW)[3] software which generates C-code from Mathworks' Matlab/Simulink block diagrams. Real-time simulation and flight control law code is generated from a symbolic model of an aircraft, subsystem or component. The overall process is represented in Figure 2.

Using RTW, the demonstration of pilot-in-the-loop flight simulation is easily integrated into the instructional content of the handling qualities classes for aeronautical engineering students. The ability to go from "pictures to code" presents exciting possibilities for academia as well as research and development oriented industries.

RTW provides the capability to generate simulation code that can run as a stand alone application. Flight control parameters are easily varied during the simulation through RTW, if the code is dynamically linked to the Matlab/Simulink environment.

Hardware and software is linked together using the RTW code generator[3]. The hardware is connected to Matlab/Simulink via the S-function input and output interface. Then, the S-function is tied to a Simulink block and used in a Simulink model. If the source code is included with the Simulink model, the RTW generator will add the hardware interface code to the model code. This provides a rapid method to create simulations or control laws that can be tested by hardware-in-the-loop simulation. This auto-coding of the simulation enables the engineering student to spend time designing the control law architecture and defining the aircraft model, rather than writing and debugging the control laws and integrating algorithms in the C-code or FORTRAN.

Aircraft handling qualities is traditionally presented as a concept with exercises in design ending with lessons analyzing control systems with

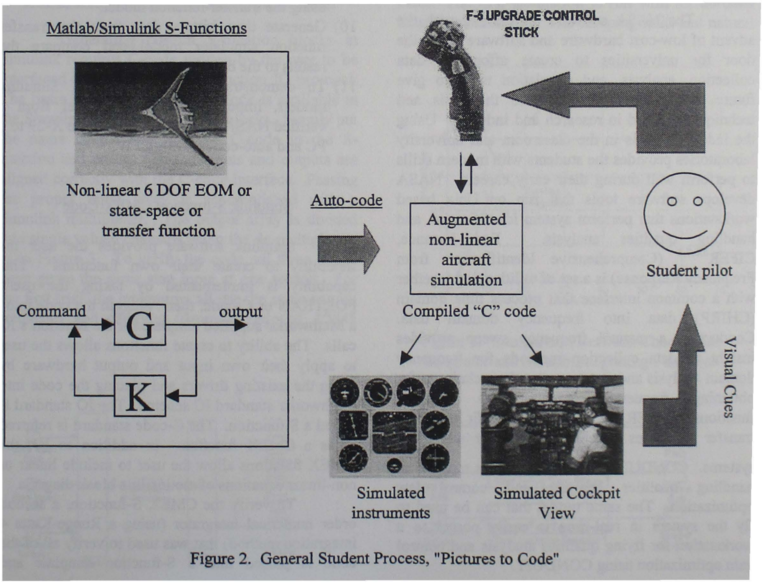


Figure 2. General Student Process, "Pictures to Code"

Computer Assisted Design (CAD) packages. Using CAD, the students generate batch time histories of their designed FWB flight control systems, but they do not interact with the design nor fly the design. The cross-coupling that many models exhibit also impairs gaining intuition of the plant, especially during the early engineering learning stages. The time required to write real-time simulation source code for control laws, aerodynamic models and the hardware interfaces for a real-time simulation with hardware distracts more from learning the fundamentals than the benefit of flying one's own design. Since RTW provides the auto-coding capability to generate real-time simulations with hardware-in-the-loop, the time required to generate the real-time simulation for each of the students' design iterations is well spent.

Non-linear simulations require no modifications to the auto-coding process. Simply include the non-linear equations of motion properly connected to the aerodynamic, gear, and thrust model.

Simulink source code is platform independent, as long as no custom blocks are included in the model. This feature has the advantage that the code can be generated on the desktop, then ported to a Unix based workstation for analysis.

The NASA software tools along with the advent of low-cost hardware and software opens the door for universities to create affordable data collection, analysis, and simulation tools to give future engineers the insight into the tools and techniques utilized in research and industry. Using the industrial tools in the classroom and university laboratories provides the students with modern skills to perform well during their early careers. NASA develops software tools that run on Unix based workstations that perform system identification and handling qualities analysis. For instance, CIFER[5,6,7,8] (Comprehensive Identification from Frequency Response) is a set of utilities tied together with a common interface that process time domain (CHIRP) data into frequency domain data. Conducting a manual frequency sweep provides insight to data collection methods for frequency domain analysis and demonstrates the techniques for obtaining frequency domain data (transfer functions). CIFER includes a utility to fit low order transfer functions to the high order identified systems. CONDUIT[2] is a set of utilities to perform handling qualities analysis and control law optimization. The same model that can be used to fly the system in real-time is easily ported to a workstation for flying qualities analysis and control gain optimization using CONDUIT.

The purpose of this paper is to demonstrate and verify the creation of linear and nonlinear models using the RTW code generator and using low-cost hardware and software developed at Cal-Poly, San Luis Obispo, along with the software tools developed at NASA Ames. The specific tasks performed are:

1) Create numerical integrators to generate baseline time domain data.
2) Create a S-function CMEX Simulink function block to demonstrate the ability to include user created functions in a Simulink model.
3) Create a block diagram in Simulink.
4) Generate time history from the Simulink model and compare to the baseline.
5) Generate C-code using the RTW code generator.
6) Generate time history data from the simulator to compare to the baseline data, to verify the code created using the user supplied function.
7) Create the same model as the S-function integrator using the built in Simulink transfer function block.
8) Generate time history data from the Simulink model and compare the results to the baseline.
9) Generate C-code using the RTW code generator using the transfer function model.
10) Generate time history data from the transfer function simulator model and compare the results to the baseline.
11) To demonstrate the portability of Simulink models, import a non-linear model from a verified NASA Simulink model of the X-29 to a PC and auto-code it with RTW.

Simulink S-function CMEX Code

Simulink software provides the user the flexibility to create their own functions. This capability is implemented by taking the users' FORTRAN or C-code, then "pluggin the code into" a Mathworks' supplied template for the function's IO calls. The ability to create functions allows the user to apply their own input and output hardware by using the existing drivers and placing the code into Mathworks' standard IO scheme. The IO standard is called a S-function. The C-code standard is referred to as a CMEX function. In addition to IO, the CMEX functions allow the user to include linear or non-linear equations of motion in a block diagram.

To verify the CMEX S-function, a second order numerical integrator (using a Runge-Kutta 4 integration method) that was used to verify all of the code is placed into a S-function template and imported into a Simulink block. The code uses three commanded angles, the input signals, and integrates each. Then, the integrated value of the input signal is outputted.

There is a built-in function which passes the time step to the internal code such that the Simulink model sets the time step for simulation. Any size integration step can be set in the Simulink environment and the integrator will use the value passed into the code. The authors use the time increment function (set to 0.5 seconds) to demonstrate the ability of a user to use existing simulation code in a Simulink block. By passing the time increment into the existing code, the new code is not limited to a hard-coded time step and can run at what ever time step the rest of the model is using. Mathworks supplies integrators that can be used by code in the S-functions, but for many users that have existing math models, they can wrap the S-function IO around the existing code and use the simulator as a block to wrap a control system around the linear or non-linear bare-airframe. Mathworks supplies a script to include all of its libraries that are required to complete the programs. The script adds the code, then compiles completed code into a 32 bit dynamic link library (dll).

Simulink Model Using User Created Function

Generating custom S-function blocks in Simulink requires C-code written by the user to be interfaced through standard S-function IO protocol. The basic Simulink S-function block is available in the Simulink library of nonlinear blocks. Simply put the name of the user defined C-code in the S-function and ensure that the inputs and outputs are aligned correctly with the C-code interface. Passing the proper input array is accomplished with a Simulink multiplexer. The output array is divided into single valued channels using the de-multiplexer, as in Figure 3. To verify the code, all three inputs were given the same step input at one second. For the first trial, the integrators all use the same natural frequency and damping (damping ratio of .12845 and natural frequency of .724017). Using a "rule of thumb," there should be about 1.3 overshoots before the system reaches a steady-state value. For the second trial, the code is modified such that each of the inputs use a different natural frequency and damping. The step signal is staggered to verify that the channels are independent and simultaneous.

Once the model is "wired up" with input and output signals, the simulation is executed.

C code Generated by "Real Time Workshop"

Once a baseline set of data is collected from the first Simulink model, the block diagram is used to generate "real time" code using the RTW' code generator. The RTW's output is a DOS .exe file. This DOS program is run and the data is compared to a baseline integrator. See Figure 4.

Baseline Transfer Function Model

The next step is to create a transfer function model of the same system as the baseline, custom S-function block, Figure 5. The values are entered into the block, the time step is set to the same value, and Runge-Kutta is selected for integration. The results are exact time histories to the baseline check-case. The first simulation is run with all the step inputs synchronized. The damping ratio and natural

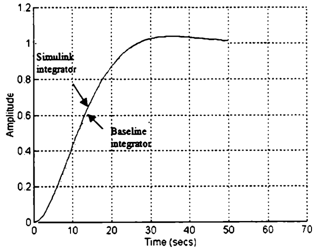


Figure 4. Comparison Between Check-Case Integrator and Matlab Integrating Functions

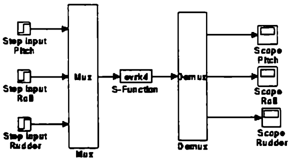


Figure 3. Simulation With User Defined S-function

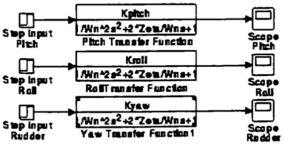


Figure 5. Simulation with Transfer Functions

frequency is set to the same values as for the check case. The second simulation is run with the step inputs staggered, like the second check-case using the custom integrator model. The transfer functions for the second simulation are set to the same values as the second check-case simulation with the custom integrators. The output data compares well between the second check-case simulation and the baseline.

RTW Code From Transfer Function Model

After the transfer function model is tested, the block diagram is run through the RTW code generator. The code is ran and the results compare to the block diagram model and the baseline.

X-29 Nonlinear Model

Finally a verified model of the NASA X-29 is obtained, ported to the PC and run through the RTW code generator. No verification is performed on the data. The model is used to confirm the portability of code between platforms. See Figure 6.

Results

The goal of writing the integrators and including them into a model is to confirm the accuracy of the code generated by RTW. Visually, the discrepancy between the time histories generated by the manually written code and the auto-coded transfer functions for first and second order transfer functions is imperceptible. The code generated by the RTW software produces the same results as the block diagram models. The code created to do the numerical integration works correctly and seamlessly as a Simulink block. The results indicate that the simulations created by Simulink on a PC are consistent and the code generated by RTW on a PC produce the same results as the Simulink but run in "real time".

Conclusions

This paper demonstrates the capabilities of desktop simulation in an academic environment. By systematic testing of the software, a procedure is established to confirm and create future simulations using Simulink and RTW. Taking existing simulation code and creating Simulink blocks is the fastest way to create real time pilot-in-the-loop simulations. The final task is to create Simulink driver blocks for the hardware-in-the-loop inceptor, instruments and graphics systems for Cal Poly's fixed base simulator. The student engineer needs only to generate the block diagrams in Simulink representing the FBW flight control system and bare-airframe model, then add the stick and instrument blocks. RTW auto-codes and compiles the C-code representing the block diagrams. The compiled auto-code is ready for immediate pilot-in-the-loop simulation. This system enables rapid design, analysis, and testing of aircraft and components for various levels of engineering students. Innovative and new aircraft can be rapidly loaded and flown in a variety of configurations. High level models of FBW flight control systems can be created and tested on an affordable desktop PC.

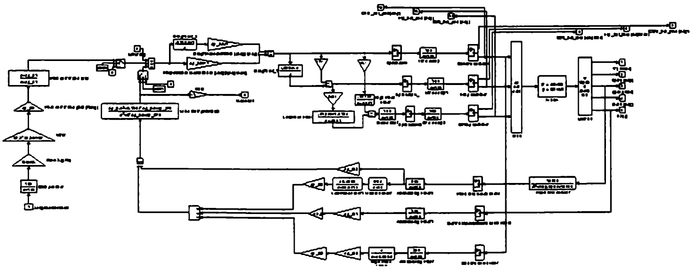


Figure 6. X-29 Flight Control System

References

1. Advances in Aircraft Flight Control, Mark B. Tischler, ed., Taylor and Frances Ltd., Bristol Pa,. 1996, pp. 35-69.

2. Tischler, M. B.; Colbourne, J. D.; Morel, M. R.; Biezad, D. J.; Levine, W. S. and Moldoveanu, V.: CONDUIT-A New Multidisciplinary Integration Environment for Flight Control Development. NASA TM-112202, USAATCOM Technical Report 97-A-009, June 1997

3. SIMULINK: Dynamic System Simulation for Matlab. The Math Works, Inc., Using Simulink, 1997.

4. SIMULINK: Real-Time Workshop. The Math Works, Inc., User's Guide, 1997.

5. Hess, R.A.: Technique for Predicting Longitudinal Pilot-Induced Oscillations. J. Guid., Cont., and Dyn., vol. 14, no. 1, 1991, pp. 198-204.

6. Chalk, C. R.: Excessive Roll Damping Can Cause Roll Ratchet. J. Guid., Cont., and Dyn., vol.6, no. 3, 1983, pp. 218-219.

7. Smith, R. E.; Sarrafian, S.K.: Effect of Time delay on Flying Qualities: An Update. J. Guid., Cont., and Dyn., vol. 9, no. 5, 1986, pp. 578-584.

8. Berry, D. T.: In Flight Evaluation of Incremental Time Delays in Pitch and Roll. J. Guid., Cont., and Dyn., vol. 9, no. 5, 1986, pp. 573-577.

9. Hess, R. A.: Effects of Time Delay on Systems Subject to Manual Control. J. Guid., Cont., and Dyn., vol. 7, no. 4, 1984, pp. 416-421.

10. Bailey, R. E.: Effect of Head-Up Display Dynamics on Fighter Flying Qualities. J. Guid., Cont., and Dyn., vol. 12, no. 4, 1983, pp. 514-520.

11. Tischler, MB; Fletcher, J. W.; Diekmann, V. L.; Williams, R. A.; Cason, R. W.: Demonstration of Frequency-Sweep Testing Technique Using a Bell 214-ST Helicopter. NASA TM-89422, USAATCOM Technical Memorandum 87-A-1, April 1987

12. Tischler, M. B.; Chen, R. T. N; The role of Modeling and Flight Testing in Rotorcraft Parameter Identification., vol. 11, no. 4, 1987, pp. 619-647

AIAA-98-4535

AIRCRAFT SIX-DEGREES-OF-FREEDOM MODEL VALIDATION THROUGH FLIGHT TEST

Shaw Y. Ong[*], Daniel Yeo[†], and Andy Gui[‡]
Singapore Technologies Aerospace Ltd
Engineering Development Centre
Singapore 539938

ABSTRACT

A six-degrees-of-freedom (6-DOF) aircraft model was formulated and a flight test program was designed to validate the flight model. Flight test data for maneuvers involving isolated pitch, roll, and yaw doublets and for coupled maneuvers as in bank-to-bank and wind-up turns were collected for model tuning and validation. Forty-five tuning parameters were used. Each is associated with a wind tunnel aerodynamic coefficient and is adjusted so that the 6-DOF aircraft flight model behaves like the real one. The validated flight model can be used for store certification work. In this paper, the complete methodology of model tuning and comparisons of simulated results with flight test data will be presented. In addition, the experiences gained in the model tuning and validation will be discussed.

INTRODUCTION

In store certification programs, the presence of a validated six degrees-of-freedom flight model of the clean aircraft is beneficial. The flight model can be used to simulate flights before actual flight tests are conducted. This is to assist in identifying potential problem areas which may be associated with the carriage of new stores within the existing flight envelope, and therefore, helps in flight test planning. It can also be used to assess quantitatively the amount of degradation in aircraft flying qualities[1] with the carriage of new stores.

As obtaining the aerodynamic data through wind tunnel tests for an upgraded aircraft is often expensive, the aerodynamic data for an earlier non-upgraded version is used. These data will be tuned accordingly with the help of flight test data until the aircraft 6-DOF flight model behaves like the actual. Two types of maneuvers were flown for flight data collection.[2-5] The first type involved isolated inputs of pitch, roll, and yaw doublets, where only a single stick input was applied. The data collected are for tuning purposes. The second type of flight maneuvers were the coupled maneuvers such as the wind-up turns, bank-to-bank, and steady heading sideslip, where a combination of roll and pitch sticks was applied. The data recorded here are for use in model acceptance and further fine tuning, if necessary.

Over two hundred flight test points were conducted for data recording; 82 for the single seater aircraft and 104 for the twin seater aircraft. The flight maneuvers covered a range of speeds from 250 to 400 KIAS and altitudes from 10,000 to 25,000 feet. The 6-DOF flight model is tuned to behave both as a single seater as well as a double seater aircraft. The primary flight variables used for model tuning and comparison with flight tests are angle-of-attack, sideslip angle, roll rate, yaw rate, pitch rate, roll angle, g-load, altitude, airspeed, elevator deflection, aileron deflection, and rudder deflection.

This paper presents the complete methodology of model tuning and provides comparisons of simulated results with flight test data. The experiences gained in the model tuning and validation will also be discussed.

METHODOLOGY

The method employed comprised tuning the wind tunnel aerodynamic data of a non-upgraded aircraft to that of the actual, upgraded, aircraft. A flow chart of the simplified 6-DOF simulation and tuning process is shown in Fig. 1. Here, the recorded pilot's stick inputs, flight condition and time flown in each test point are entered into the simulation. The simulation model trims the aircraft at the initial flight condition before going into the simulation loop. The outputs from the simulation loop are then compared with the flight test data. If simulated results do not match the flight test data, the aerodynamic model is tuned.

[*] Assistant Principal Engineer, Member AIAA
[†] Aeroscience Engineer
[‡] Aeroscience Engineer, Member AIAA

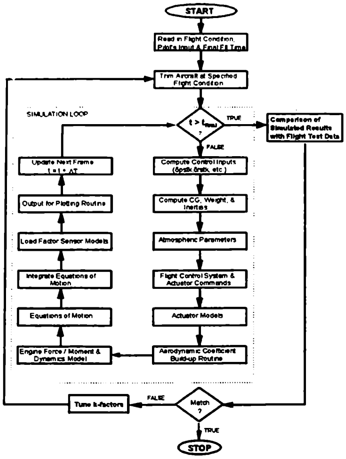

START
Read in Flight Condition, Pilot's Input & Final Flt Time
Trim Aircraft at Specified Flight Condition
SIMULATION LOOP
$t > t_{final}$?
TRUE
FALSE
Comparison of Simulated Results with Flight Test Data
Update Next Frame $t = t + \Delta T$
Compute Control Inputs (δpstk, δrstk, etc.)
Output for Plotting Routine
Compute CG, Weight, & Inertias
Load Factor Sensor Models
Atmospheric Parameters
Integrate Equations of Motion
Flight Control System & Actuator Commands
Equations of Motion
Actuator Models
Engine Force / Moment & Dynamics Model
Aerodynamic Coefficient Build-up Routine
Tune k-factors
FALSE
Match ?
TRUE
STOP

Fig. 1 Simplified 6-DOF simulation and tuning flow chart

Tuning Process

The aerodynamic tuning process involves a total of forty-five parameters (k-factors). Each of these is associated with a wind tunnel aerodynamic coefficient and is adjusted so that the 6-DOF aircraft flight model behaves like the real one.

To illustrate this process, consider the total pitching moment coefficient, *Cm*, of the aircraft:

$$\begin{aligned} Cm = {} & k_1 * Cm_0 + k_2 * Cm_{dh} * dh + k_3 * Cm_{de} * de \\ & + k_4 * Cm_q * q * \frac{c}{2v} + k_5 * Cm_{\dot\alpha} * \dot\alpha * \frac{c}{2v} \\ & + k_6 * Cm_\alpha * \alpha \end{aligned}$$

The aerodynamic coefficients Cm_0, Cm_{dh}, Cm_{de}, Cm_q, $Cm_{\dot\alpha}$, and Cm_α are wind tunnel data of an earlier non-upgraded version of the aircraft. The *k's* are the tuning parameters which will be adjusted so that the simulated results match flight test data. Similar effort is applied to the k-factors in total lift coefficient C_L, drag coefficient C_D, side force coefficient C_Y, rolling moment coefficient C_l, and yawing moment coefficient C_n. This tuning process continues until all flight test points have been tested and simulated results compare favorably with all flight test data. Briefly, the tuning process is as follows:

Steps

1. Use actual pilot's stick inputs from flight test to drive the 6-DOF flight simulation model.
2. Compare simulated results with flight test data.
3. If simulated results match favourably with flight test data, stop.
4. If not, tune k-factors. Go to Step 2.

It was found that the k-factors that contributed to significant changes in the simulation results were those associated with the damping coefficients in the aerodynamic data. These damping coefficients are the pitching moment coefficient due to pitch rate, the rolling moment coefficient due to roll rate, and the yawing moment coefficient due to yaw rate. The k-factors associated with the control power derivatives also caused significant changes in the simulation results when they were adjusted.

Two separate sets of k-factors were obtained - one for the single seater and one for the two seater aircraft. These sets of k-factors are applicable to all maneuvers.

RESULTS

A large number of curves were plotted and analyzed. However, only plots for three test points are given here. They are representative of the trend followed by similar plots for other test points. The three plots show the behaviour of the tuned flight model in more difficult maneuvers. These are: a bank-to-bank maneuver, a double 360° roll maneuver (clockwise roll followed by anti-clockwise roll) and a steady heading sideslip maneuver.

Bank-to-Bank Maneuver

This maneuver involves multi-control stick inputs, i.e., both roll and pitch sticks are used. In this maneuver, the aircraft is rolled to one side, pitched up, and then roll reversed to the other side while simultaneously unloading the pitch stick. The entry speed and altitude are 300 kts and 15,000 ft, respectively. Figs. 2(a-h) compare the various simulated flight parameter time histories with the flight test data. The solid curve refers to the flight test data while the dash and the dash-dot curves refer to the tuned and untuned simulated results, respectively.

It can be seen from these plots that the flight parameter histories of the tuned flight model match the flight test data significantly better than those of the untuned model. This is particularly true for the roll and yaw rates (see Figs. 2(e, f)). The altitude and velocity profiles are not as accurate in the comparison with the flight test data, however. Nonetheless, the tuned model still gives far better results than the untuned model as shown in

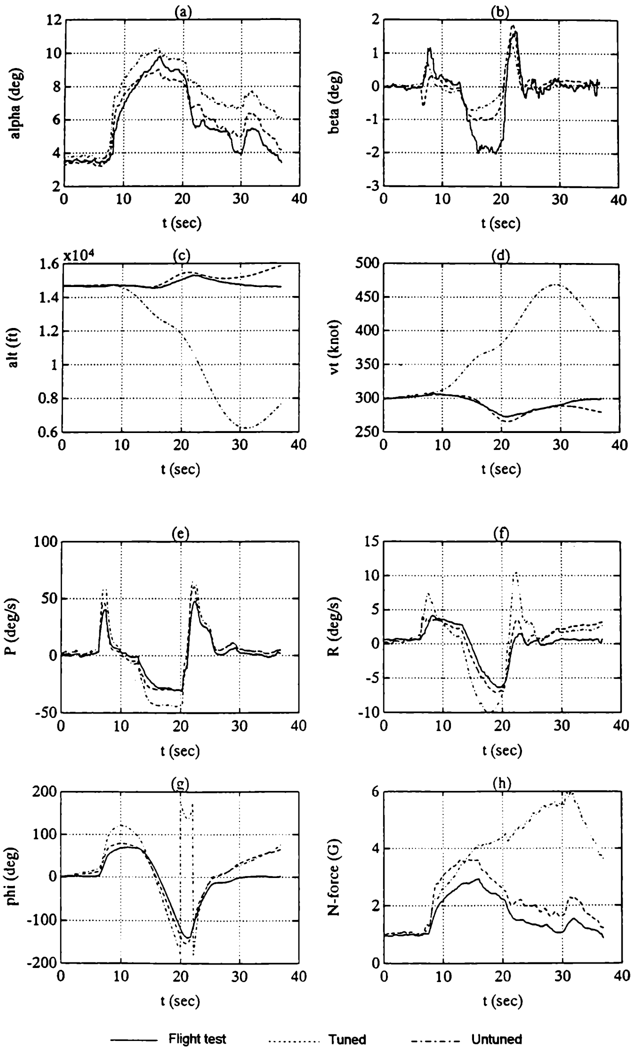


Fig. 2(a-h). Comparison of tuned and untuned flight model results with flight test data for aircraft bank-to-bank maneuver

Figs. 2(c, d). A similar trend is seen in the angle-of-attack and normal load factor histories.

360° Roll Maneuver

In this maneuver, the aircraft is initially trimmed for straight and level flight at Mach 0.7 at 20,000 ft. A ¼ aileron stick is then applied for a 360° roll. The aircraft is retrimmed and a reverse ¼ aileron stick is applied until a reverse 360° roll is achieved. The actual aileron roll stick history for this maneuver is shown in Fig. 3 and was applied to the simulation model.

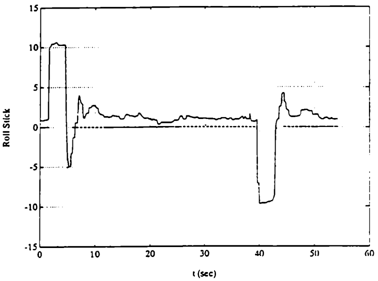


Fig. 3 Aileron roll stick input history for 360° roll

Figs. 4(a-h) show a comparison of the various simulated flight parameter time histories with the flight test data. The solid curve refers to the flight test data while the dash and the dash-dot curves refer to the tuned and untuned simulated results, respectively.

Again it can be seen from these plots that the tuned flight model performed significantly better than the untuned model in matching its flight parameter histories with the flight test data. This is clearly reflected in the roll rates and the roll angle histories (see Figs. 4(e, g)), where the tuned model produced almost identical results to the flight test. The roll angle of the untuned model had strayed away. The altitude and velocity profiles of the tuned model also match very well with that of the flight test data compared to the untuned model. The oscillations in the angle-of-attack, sideslip, yaw rate, and normal load factor histories seen in the simulation results (Figs. 4(a,b,f,h)) are the transient responses due to instantaneous slats deployment in the flight model. The actual aircraft slats deployment had been gradual, and thus, there were no observable transient responses.

Steady Heading Sideslip

In this maneuver, the aircraft is trimmed for straight and level flight at 200 knots at 20,000 ft. A right ½ yaw pedal is applied to achieve a steady sideslip followed by a left ½ yaw pedal whilst maintaining the same heading. The actual yaw pedal history used in the simulation model for this maneuver is shown in Fig. 5. The primary goal is to compare the simulated sideslip history with the flight test data.

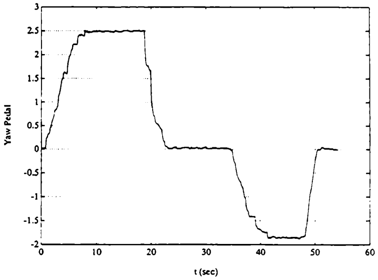


Fig. 5 Yaw pedal input history for steady heading sideslip

Figs. 6(a-h) show the comparison of the various simulated flight parameter time histories with the flight test data. The solid curve refers to the flight test data while the dash and the dash-dot curves refer to the tuned and untuned simulated results, respectively.

The sideslip histories of the tuned and the untuned model (see Fig. 6b) show that both models behave closely to that of the actual aircraft. The absolute sideslip magnitudes of the flight model, however, are slightly lower than that of the actual aircraft. This is because the flight model's rudder deflection is slightly smaller than that of the actual aircraft. In altitude, velocity, roll angle, and normal load factor history comparisons, however, the tuned flight model performed significantly better than the untuned model. This is clearly shown in Figs. 6 (c,d,g,h), respectively. The oscillations seen in the roll and yaw rates of Figs. 6(e, f) show that the aircraft's motion had been jittery when the maneuver was executed. This naturally led to difficulties in the angle-of-attack history comparison.

MODEL TUNING EXPERIENCES

The process of tuning a flight model to match the flying behaviour of an actual aircraft can be tedious and arduous. There are several points worth

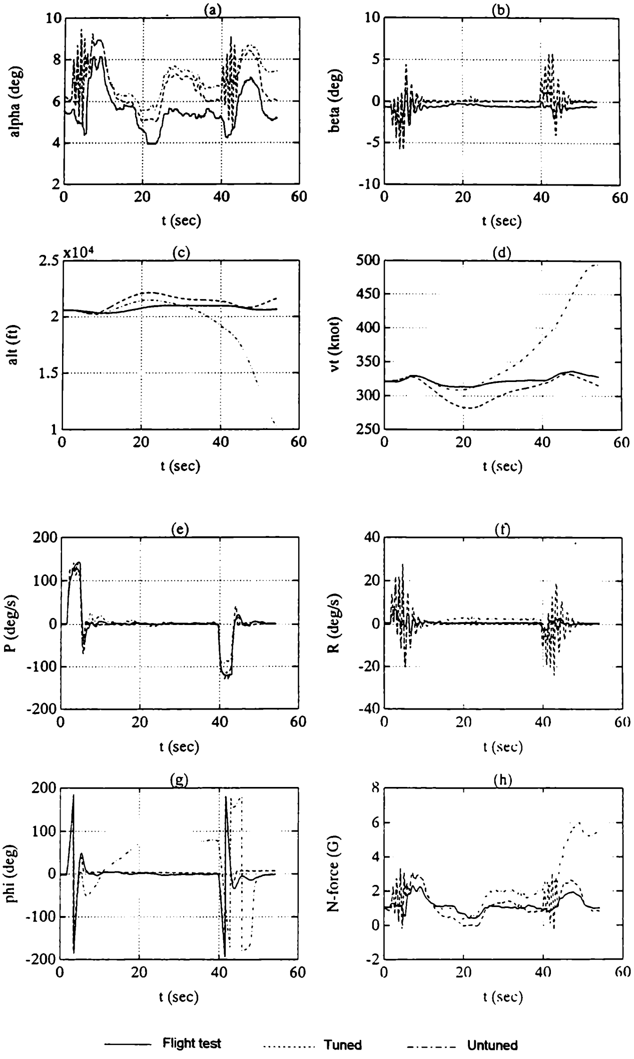


Fig. 4(a-h). Comparison of tuned and untuned flight model results with flight test data for aircraft 360° roll maneuver

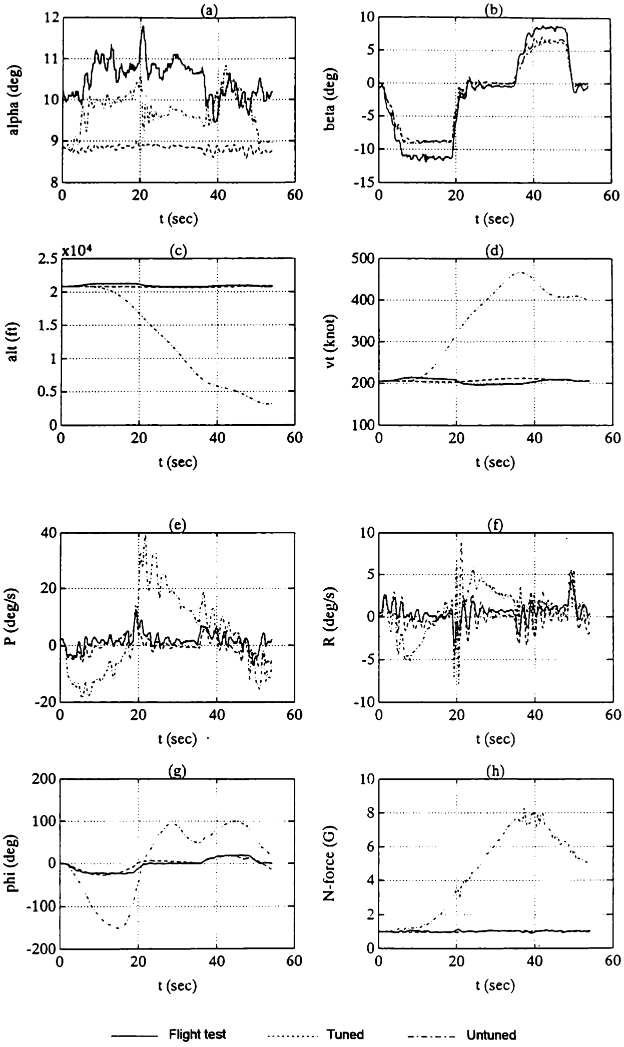


Fig. 6(a-h). Comparison of tuned and untuned flight model results with flight test data for aircraft steady heading sideslip maneuver considering when doing this, and these will provide a better understanding of the potential tuning difficulties.

1. There will likely be a lot of fluctuations, measurement noise, and data dropouts in the collected flight data from the telemetry system, including in the recorded pilot's control stick inputs. As the control stick input data from flight test are used to drive the simulation model, it is important that noise and spikes due to data dropouts be smoothened before use in the flight model. Erroneous aircraft reponse will result if this is not done.
2. In the response matching process, all recorded, and filtered, control stick inputs are used in the flight model for simulation. This can lead to unmatched responses. For example, if a crosswind occurs in a flight test, the pilot will use aileron deflection to maintain straight and level flight without roll rate. However, as there is no wind modeling in the flight model, this aileron roll stick input will create roll rates in the flight model, leading to poor response matching. It is, therefore, important that disturbances such as wind, which are not modeled in the flight model, be kept to a minimum when collecting flight data.
3. The initial control stick values obtained from the flight test data, and used to start the model simulation are crucial. The simulation model used always treats the initial control stick values as trimmed values even if they are not. And if the flight control sticks are not trimmed, subsequent simulation results can deviate drastically from the flight test data. The flight model used has a built-in trimming routine that will trim all translational accelerations and rotational rates to zero and thus always begin from a trimmed condition, which may not always be true in actual flight. The trimmed condition is particularly important for matching of longitudinal dynamics represented by time histories of angle-of-attack, magnitudes of the velocity, altitude, and pitch rate.
4. It is important to know all instrumentation settings, how the measured data were calibrated, and the accuracy and limitations of each instrument. In particular, the sign convention used in the measured data should be consistent with that used in the simulation model. Bias can also exist in flight test data which must be accounted for during the tuning process.

CONCLUSIONS

A six degrees-of-freedom simulation model has been constructed and successfully tuned and validated through flight test. This was made possible through the use of adjustable *k*-factors assigned to each aerodynamic term forming the total aerodynamic coefficients. Results have shown that the aerodynamically tuned flight model behaves closely to the actual aircraft. This model can be used in store certification programs to analyze the behavior of the aircraft carrying new stores, before conducting flight tests. By identifying potential problem areas, before flight tests are flown, risks can be minimized.

ACKNOWLEDGEMENTS

This work was conducted under contract with the DSO National Laboratories. The first author wishes to thank Dr. Lou Kok Yong and Patrick Teo of DSO National Laboratories for the many helpful discussions in the project.

REFERENCES

1. Anon., *MIL-F-8785C, Military Specification Flying Qualities of Piloted Airplanes*, Air Force Flight Dynamics Laboratory, WPAFB, Dayton, Ohio, November 1980.
2. Ward, D. T., *Introduction to Flight Test Engineering*, Kendall/Hunt, Dubuque, Iowa, 1996.
3. Anon., *Pilots Handbook for Critical and Exploratory Flight Testing*, The Society of Experimental Test Pilots, Lancaster, California, and AIAA, 1972.
4. Roberts, S. C., *Light Aircraft Performance for Test Pilots and Flight Test Engineers*, Lecture Notes, Flight Research, Inc., Mojave, California, 1987.
5. Roberts, S. C., *Flying Qualities Flight Testing of Light Aircraft for Test Pilots and Flight Test Engineers*, Lecture Notes, Flight Research, Inc., Mojave, California, 1987.

AIAA-98-4536

INTEGRATED PROJECTION GUIDANCE AND NONLINEAR CONTROL OF UNMANNED AIR VEHICLE

Yiing-Yuh Lin* and Sern-Sih Liu†
Department of Aeronautics and Astronautics
National Cheng Kung University
Tainan, Taiwan, R.O.C.
E-Mail: fcl6yyln@mail.iaa.ncku.edu.tw

Abstract

In this study, an integrated projection guidance and nonlinear flight control law (IPGNFC) is proposed for the unmanned air vehicle (UAV). Testing procedures of the IPGNFC and the results of the testing are reported. To realistically evaluate the design and to compare the results of different types of simulation, the simulation was done on nonreal-time PC and the real-time two PC platform. The two PC in the platform are connected by AD/DA interface cards and can run in parallel.

For designing the IPGNFC, the control law can be divided into the guidance and the flight control problem. The guidance problem, based on the point mass model, is to keep the air vehicle stay on a flight trajectory toward the target location with constant air speed. Instead of using the traditional gain scheduling method, the flight controller is designed with nonlinear feedback theory. By the use of singular perturbation, the aircraft system dynamics is separated into fast and slow model. Then feedback linearization and sliding mode control were used to design a robust flight controller. A test case is included which requires the proposed IPGNFC to lead an UAV to arrive at a preset target position and to perform a circling maneuver.

Introduction

Unmanned air vehicle is one of the small but rapidly growing areas in the aerospace engineering[1]$forresearchanddevelopment$. Like the robots on the ground or the satellites in space, the main research objectives of the UAV are to achieve full automation of the vehicle and to perform required operation under all kinds of environmental condition. Although UAV has been with military related applications for more than twenty five years, there are still many problems warrant detailed and up-to-date analysis. Among them is the autopilot design with new technologies. The advent of modern control theories, onboard electronics, and new INS/GPS navigation system allow the controller to be more flexible and powerful than it used to be.

For common practice, the command given to an UAV may be flying from an initial location through a number of way points and performs some predefined maneuver when the final location is reached. Conventional small perturbation design approach or the gain scheduling method is not efficient for designing such flight controller. Several advanced flight controller design methods including the model following[2], the nonlinear inverse dynamics[3], the singular perturbation[4] have been proposed. In those papers, the control of angle of attack, sideslip angle, and velocity were the main purposes.

In this paper, the objective of the control is to coordinate the control effectors to achieve a commanded guidance trajectory. The approach proposed in this paper first separates the dynamics of the aircraft into fast and slow subsystems. Lie transform is then applied to formulate the transformation coordinates which in turn linearize the two sets of nonlinear subsystem equations. An integrated guidance and flight controller is formulated using sliding mode control method. The performance of the controller demonstrated through an example has shown that the commanded guidance can be achieved with flight robustness.

Input/Output Feedback Linearization

In this section, we review Lie transform technique which linearizes the system equations through input/output relations. This technique was used to design the flight controllers in the following sections.

Consider the nonlinear system on R^n

$$\dot{x} = f(x) + G(x)U \quad (1)$$
$$y = h(x) \quad (2)$$

*Associate Professor, AIAA member
†Graduate Student
‡

where $f(x) \in R^n$, $G(x) \in R^{n\times m}$ are smooth function and smooth vector field. $U \in R^m$ is the control input and $h(x) \in R^m$ is the output function. Let r_i be the relative degree of y_i, and through Lie derivatives[5,6] we obtain

$$y_i^{r_i} = L_f^{r_i} h_i + \sum_{j=1}^{m} L_{G_j} L_f^{(r_i-1)} h_i U_j \tag{3}$$

where $L_{G_j} L_f^{(r_i-1)} h_i \neq 0$ for at least one j for x, and G_j the $j-th$ column of $G(x)$. From Eq.(3), we obtain the input-output relations as

$$\begin{bmatrix} y_1^{r_1} \\ y_2^{r_2} \\ \vdots \\ y_m^{r_m} \end{bmatrix} = F + \hat{G}(x)U \tag{4}$$

where

$$F = \begin{bmatrix} L_f^{r_1} h_1(x) \\ L_f^{r_2} h_2(x) \\ \vdots \\ L_f^{r_m} h_m(x) \end{bmatrix} \quad \text{and}$$

$$\hat{G}(x) = \begin{bmatrix} L_{G_{11}} & L_{G_{21}} & & L_{G_{m1}} \\ L_{G_{12}} & L_{G_{22}} & & L_{G_{m2}} \\ \vdots & \vdots & & \vdots \\ L_{G_{1m}} & L_{G_{2m}} & & L_{G_{mm}} \end{bmatrix}$$

where $L_{G_{ij}} = LG_i L_f^{r_j-1} h_j(x)$. If the decoupling matrix $\hat{G}(x)$ is invertible, the input-output relationship can be linearized and decoupled by choosing U as

$$U = \hat{G}(x)^{-1} \begin{bmatrix} \nu_1 - L_f^{r_1} h_1(x) \\ \nu_2 - L_f^{r_2} h_2(x) \\ \nu_3 - L_f^{r_3} h_3(x) \\ \vdots \\ \nu_m - L_f^{r_m} h_m(x) \end{bmatrix} \tag{5}$$

where $\nu = col(\nu_1 \ \ \nu_2 \ \ \cdots \ \ \nu_m)$ is a set of commands to the system. The system can be controlled like a linear system and tracks the command ν perfectly.

The system is said to have total relative degree $r = \sum_{i=1}^{m} r_i$, where $r \leq n$. If the total relative degree is equal to the system order, $r = n$, an invertible decoupling matrix is always exists and the stability of the system by input-output linearization is guaranteed. However, if $r < n$, unobservable states dynamics will exist after input-output linearization and the stability can not be guaranteed unless all the hidden states dynamics[5] are asymptotically stable. To ensure $r = n$, the technique 'dynamic extension' proposed by Isidori[6] is used in this research which includes the derivatives of some system inputs as extra inputs and allows $\hat{G}(x)$ to be invertible.

Guidance Via Sliding Mode Control

The nonlinear system, as derived in the previous section, can be transformed into linear system by choosing an appropriate set of input functions to cancel the nonlinear terms. However, the input-output linearization is valid only for very accurately modeled systems. To guarantee the control robustness in the presence of system uncertainties, such as the parameter uncertainty or unmodeled dynamics, the sliding mode control method is applied to the resultant linear equations. Consider the nonlinear system model (1,2). By the input-output linearization, the system equations with uncertainties Δf and $\Delta\hat{G}(x)$ become

$$\begin{bmatrix} y_1^{r_1} \\ y_2^{r_2} \\ \vdots \\ y_m^{r_m} \end{bmatrix} = F + \Delta f + [\hat{G}(x) + \Delta\hat{G}(x)]U \tag{6}$$

where the uncertainties $\parallel \Delta f \parallel$ and $\parallel \Delta\hat{G}(x) \parallel$ are bounded. We can select the switching surface as

$$S_i = (\frac{d}{dt} + \lambda_i)^{r_i-1}(y_i - y_{id}) \tag{7}$$

where λ_i is a positive constant and y_{id} is the desired response. Differentiating Eq.(7) yields

$$\dot{S}_i = y_i^{r_i} + \sum_{j=0}^{r_i-1} c_{ij}(y_i^j - y_{id}^j) - y_{id}^{r_i} \tag{8}$$

To design a robust controller, let

$$U = \hat{G}(x)^{-1} \left\{ \begin{bmatrix} y_{1d}^{r_1} \\ y_{2d}^{r_2} \\ \vdots \\ y_{md}^{r_m} \end{bmatrix} - F - CY - Ksign(S) \right\}$$

where

$$CY = \begin{bmatrix} \sum_{j=0}^{r_1-1} c_{1j}(y_1^j - y_{1d}^j) \\ \sum_{j=0}^{r_2-1} c_{2j}(y_2^j - y_{2d}^j) \\ \vdots \\ \sum_{j=0}^{r_m-1} c_{mj}(y_m^j - y_{md}^j) \end{bmatrix}$$

Substituting Eq.(6) and control U into Eq.(8), one obtain

$$\begin{aligned} \dot{S} &= \Delta f - Ksign(S) \\ &\quad - \Delta\hat{G}(x)\hat{G}(x)^{-1}[CY + Ksign(S)] \end{aligned} \tag{9}$$

For the guidance problem to be asymptotically stable, we let $S_i\dot{S}_i < 0$ or $S_i\dot{S}_i < -\eta \parallel S_i \parallel$, where η is a small positive constant, and[7]

$$\parallel K_i \parallel \geq \frac{\eta + \parallel \Delta f_i \parallel + \parallel \Delta\hat{G}(x)\hat{G}(x)^{-1} CY \parallel}{1 - \parallel \Delta\hat{G}(x)\hat{G}(x)^{-1} \parallel}$$

Dynamics of Aircraft

As defined in figure 1, the equations of motion of a aircraft expressed in terms of body fixed coordinates is a set of nonlinear differential equations[3] which can be written as

$$\dot{u} = \frac{F_x}{m} - qw + rv \tag{10}$$

$$\dot{v} = \frac{F_y}{m} - ru + pw \tag{11}$$

$$\dot{w} = \frac{F_z}{m} - pv + qu \tag{12}$$

$$\dot{p} = \frac{M_x}{I_{xx}} - \frac{I_{zz} - I_{yy}}{I_{xx}} qr + \frac{I_{xz}}{I_{xx}}(\dot{r} + pq) \tag{13}$$

$$\dot{q} = \frac{M_y}{I_{yy}} - \frac{I_{xx} - I_{zz}}{I_{yy}} rp - \frac{I_{xz}}{I_{yy}}(p^2 - r^2) \tag{14}$$

$$\dot{r} = \frac{M_z}{I_{zz}} - \frac{I_{yy} - I_{xx}}{I_{zz}} pq - \frac{I_{xz}}{I_{zz}}(qr - \dot{p}) \tag{15}$$

$$\dot{\phi} = p + q \tan\theta \sin\phi + r \tan\theta \cos\phi \tag{16}$$

$$\dot{\theta} = q \cos\phi - r \sin\phi \tag{17}$$

$$\dot{\psi} = r \cos\phi \sec\theta + q \sin\phi \sec\theta \tag{18}$$

We may also transform Eqs.(10-12) from body fixed coordinates to wind coordinates, which give

$$\begin{aligned} \dot{V} =& \frac{F_x}{m} \cos\alpha \cos\beta + \frac{F_y}{m} \sin\beta \\ &+ \frac{F_z}{m} \sin\alpha \cos\beta \end{aligned} \tag{19}$$

$$\begin{aligned} \dot{\alpha} =& \frac{1}{mV \cos\beta}(F_z \cos\alpha - F_x \sin\alpha) + q \\ &- \tan\beta(p \cos\alpha + r \sin\alpha) \end{aligned} \tag{20}$$

$$\begin{aligned} \dot{\beta} =& \frac{\cos\beta}{mV} F_y + p \sin\alpha - r \cos\alpha \\ &- \frac{\sin\beta}{mV}(F_z \sin\alpha + F_x \cos\alpha) \end{aligned} \tag{21}$$

where F_i and M_i, $i = x, y, z$, are the aerodynamic forces and moments. Consider five control inputs to design the control law. They are the elevator δ_e, the aileron δ_a, the rudder δ_r, the flaperon δ_{lef}, and the engine throttle η.

Design of Fast-Mode Controller

According to flight conditions, we consider the rotational motions usually are faster than the translational motions. An implicit robust model following with full states feedback controller is designed for the fast dynamics by the sliding-mode control method. The dynamic equations of the fast states can be expressed as[8,9]

$$\dot{p} = F_p + P_{11}\delta_a + P_{12}\delta_r \tag{22}$$

$$\dot{q} = F_q + Q_{11}\delta_e + Q_{12}\delta_{lef} \tag{23}$$

$$\dot{r} = F_r + R_{11}\delta_a + R_{12}\delta_r \tag{24}$$

where

$$\begin{aligned} F_p =& Apq + Bqr + \{C(C_{Nb} + C_{N\beta}\beta) \\ &+ D(C_{Lb} + C_{L\beta}\beta)\}\bar{q}S\bar{b} \\ F_q =& Epr + F(r^2 - p^2) \\ &+ G(C_{Mb} + C_{M\alpha}\alpha + C_{Mq}q)\bar{q}S\bar{c} \\ F_r =& Hqr + Ipq + \{J(C_{Nb} + C_{N\beta}\beta) \\ &+ D(C_{Lb} + C_{L\beta}\beta)\}\bar{q}S\bar{b} \\ P_{11} =& \bar{q}S\bar{b}(CC_{N\delta_a} + DC_{L\delta_a}) \\ P_{12} =& \bar{q}S\bar{b}(CC_{N\delta_r} + DC_{L\delta_r}) \\ Q_{11} =& \bar{q}S\bar{c}(GC_{m\delta_e}), \quad Q_{12} = \bar{q}S\bar{c}(GC_{m\delta_{lef}}) \\ R_{11} =& \bar{q}S\bar{b}(JC_{N\delta_a} + DC_{L\delta_a}) \\ R_{12} =& \bar{q}S\bar{b}(JC_{N\delta_r} + DC_{L\delta_r}) \\ A =& \frac{I_{xz}(I_{xx} + I_{zz} - I_{yy})}{I_{xx}I_{zz} - I_{xz}^2} \\ B =& \frac{I_{yy}I_{zz} - I_{zz}^2 - I_{xz}^2}{I_{xx}I_{zz} - I_{xz}^2} \\ C =& \frac{I_{zz}}{I_{xx}I_{zz} - I_{xz}^2}, \quad D = \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} \\ E =& \frac{I_{zz} - I_{xx}}{I_{yy}}, \quad F = \frac{I_{xz}}{I_{yy}}, \quad G = \frac{1}{I_{yy}} \\ H =& \frac{I_{xz}(I_{yy} - I_{zz} - I_{xz})}{I_{xx}I_{zz} - I_{xz}^2} \\ I =& \frac{I_{xz}^2 + I_{xx}^2 - I_{xx}I_{yy}}{I_{xx}I_{zz} - I_{xz}^2} \\ J =& \frac{I_{xx}}{I_{xx}I_{zz} - I_{xz}^2} \end{aligned}$$

Let the fast system track a stable model

$$\dot{p}_d = \tau_p(p_c - p_d) \tag{25}$$

$$\dot{q}_d = \tau_q(q_c - q_d) \tag{26}$$

$$\dot{r}_d = \tau_r(r_c - r_d) \tag{27}$$

where (p_d, q_d, r_d) are the model states, (τ_p, τ_q, τ_r) are the constant coefficients to be determined for the model, and (p_c, q_c, r_c) are the attitude commands. We choose the switching function to be

$$S = \begin{bmatrix} p - p_d \\ q - q_d \\ r - r_d \end{bmatrix} \tag{28}$$

To avoid chattering of control efforts caused by the sign function, boundary thickness(ϵ) is used instead of sign function in Eq.(9) to relax the saturation of control signals

$$sat(s/\epsilon) = \begin{cases} 1 & \text{if } s/\epsilon > 1 \\ s/\epsilon & \text{if } -1 \le s/\epsilon \le 1 \\ -1 & \text{if } s/\epsilon \le -1 \end{cases}$$

where

$$G_{3\times 4} = \left[\begin{array}{llll} -a\tan\beta\cos\alpha & \frac{F_z}{m}+a & g\sin\phi\cos\theta-\frac{F_y}{m} & \frac{K_\eta T_{max}}{m} \\ & -g\cos\theta\cos\phi & -a\tan\beta\sin\alpha & \\ -\frac{F_z}{m}+b\sin\alpha & 0 & \frac{F_x}{m}+g\sin\theta & 0 \\ +g\cos\phi\cos\theta & & -b\cos\alpha & \\ \frac{F_y}{m}-c\tan\beta\cos\alpha & -\frac{F_x}{m}+c-g\sin\theta & -c\tan\beta\sin\alpha & 0 \\ -g\sin\phi\cos\theta & & & \end{array}\right]$$

The control functions are chosen to be

$$\begin{bmatrix} 0 & 0 & P_{11} & P_{12} \\ Q_{11} & Q_{12} & 0 & 0 \\ 0 & 0 & R_{11} & R12 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{lef} \\ \delta_a \\ \delta_r \end{bmatrix}$$

$$= \begin{bmatrix} -F_p + \tau_p(p_c - p_d) - K_p sat(S_1/\epsilon_p) \\ -F_q + \tau_q(q_c - q_d) - K_q sat(S_2/\epsilon_q) \\ -F_r + \tau_r(r_c - r_d) - K_r sat(S_3/\epsilon_r) \end{bmatrix}$$

Due to physical constraints on the control surface deflections, a minimum norm solution is used for limiting the control surface deflections[9].

Design of Slow-Mode Controller

For guidance purpose, a transformation between the body-fixed coordinates and the inertial coordinates for the translational velocity of the UAV can be defined as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = [T] \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \tag{29}$$

where $[T]$ is the (3-2-1) transformation matrix.

The transient dynamics of the fast states have negligible effect on the slow states. The fast states (p, q, r) and the throttle η are used as the control variables for the slow subsystem. Therefore, the slow subsystem of the aircraft can be expressed in term of nine state equations $(V, \alpha, \beta, u, v, w, \phi, \theta, \psi)$ without the angular rates dynamics. To design the nonlinear controller for slow subsystem, there are four control variables and only four independent output variables can be controlled at most. (X, Y, Z) are chosen as the output variables. A fourth output is chosen to be v so that the side slip angle is controlled near zero. The output function was repeatedly differentiated until at least one control variable appeared. To avoid fast change of the thrust the throttle is defined as

$$\dot{\eta} = K_\eta(\eta_c - \eta) \tag{30}$$

where K_η is a small positive constant and η_c is the throttle command which is used as a new control variable. The total relative degree is now equal to the system order and no internal dynamics exist. differentiating Eq.(29) yields[8]

$$\frac{d^3}{d^3t}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [T]^T\{\begin{bmatrix} f_{xx} \\ f_{yy} \\ f_{zz} \end{bmatrix} + G_{3\times 4}\begin{bmatrix} p \\ q \\ r \\ \eta_c \end{bmatrix}\}$$

$$\dot{v} = pw - ru + \frac{F_y}{m} \tag{31}$$

$$\begin{aligned} f_{xx} &= \frac{a}{mV\cos\beta}(F_z\cos\alpha - F_x\sin\alpha) \\ &+ \frac{\rho V\dot{V}}{m}SC_x - \frac{K_\eta T_{max}}{m}\eta \\ f_{yy} &= \frac{b\cos\beta}{mV}F_y - \frac{b\sin\beta}{mV}(F_z\sin\alpha + F_x\cos\alpha) \\ &+ \frac{\rho V\dot{V}}{m}SC_y \\ f_{zz} &= \frac{c}{mV\cos\beta}(F_z\cos\alpha - F_x\sin\alpha) \\ &+ \frac{\rho V\dot{V}}{m}SC_z \\ a &= \frac{\bar{q}SC_{x\alpha}}{m}, \quad b = \frac{\bar{q}SC_{y\beta}}{m}, \quad c = \frac{\bar{q}SC_{z\alpha}}{m} \end{aligned}$$

If the desired dynamics of the output variables are specified as

$$\frac{d^3X_d}{dt} = \tau_x(a_x - \ddot{X}_d) \tag{32}$$

$$\frac{d^3Y_d}{dt} = \tau_y(a_y - \ddot{Y}_d) \tag{33}$$

$$\frac{d^3Z_d}{dt} = \tau_z(a_z - \ddot{Z}_d) \tag{34}$$

$$\dot{v}_d = -\tau_v v_d \tag{35}$$

where $(\tau_x, \tau_y, \tau_z, \tau_v)$ are positive constants and (a_x, a_y, a_z) are the acceleration commands. the third order dynamics on (X, Y, Z) can be thought as that the acceleration of the six DOF aircraft is different from that of point mass model by the first order lag due to the actuator dynamics. Also by the sliding mode control method, we define the switching function as

$$S = \begin{bmatrix} (\ddot{X} - \ddot{X}d) + 2\lambda_x(\dot{X} - \dot{X}_d) + \lambda_x^2(X - X_d) \\ (\ddot{Y} - \ddot{Y}d) + 2\lambda_y(\dot{Y} - \dot{Y}_d) + \lambda_y^2(Y - Y_d) \\ (\ddot{Z} - \ddot{Z}d) + 2\lambda_z(\dot{Z} - \dot{Z}_d) + \lambda_z^2(Z - Z_d) \\ v - v_d \end{bmatrix}$$

Design of Projection Guidance

To assist an UAV to arrive at a target location, the projection guidance method is developed in the research. The method takes in the present position of the UAV and the target location, calculates the guidance direction to continuously correct the flight path toward final target. The guidance command derived becomes the command of the slow mode flight controller. To design the projection guidance law considers 1. the guidance law will be in the form of acceleration, 2. in the course of change direction the speed of the airplane shall not vary too much, 3. the direction change shall be done in shortest time possible. For guidance purpose, one is necessary to represent the velocity of the UAV in inertial coordinate (X, Y, Z).

Flying to a Way Point

In reference to figure 2, to derive the projection guidance, the unit vector $\vec{U}_{pn} = (a_{xn}, a_{yn}, a_{zn})$ of the guidance command $\vec{U}_p$ is determined from the position vector $\vec{R}_{tp} = (X, Y, Z)$ from the UAV to the way-point, the velocity vector $\vec{V}_p = (\dot{X}_p, \dot{Y}_p, \dot{Z}_p)$ of the UAV, and the unit vector $\vec{e}_n = (e_1, e_2, e_3)$ perpendicular to the plane defined by $\vec{R}_{tp}$ and $\vec{V}_p$ with

$$\vec{e}_n = \frac{\vec{V}_p}{|\vec{V}_p|} \times \frac{\vec{R}_{tp}}{|\vec{R}_{tp}|} \tag{36}$$

$$\vec{U}_{pn} = \vec{e}_n \times \frac{\vec{V}_p}{|\vec{V}_p|} \tag{37}$$

Once we find $\vec{U}_{pn}$, we can determine the magnitude $|\vec{U}_p|$ by

$$|\vec{U}_p| = \zeta \theta_i U_{pmax} (1 - \frac{\theta_i - \theta}{\theta_i} e^{-\kappa\theta}) \tag{38}$$

where ζ is the command magnifying constant, θ_i is the initial angle between $\vec{V}_p$ and $\vec{R}_{tp}$, U_{pmax} is the maximum thrust of UAV, θ is the angle between $\vec{V}_p$ and $\vec{R}_{tp}$, κ is the command curve shaping parameter. Eq.(38) will guide the $\vec{V}_p$ to overlap with $\vec{R}_{tp}$ with little overshoot.

Horizontal Circling

When the way point is reached, the UAV is then switched to circling mode with similar guidance strategy described previously. The speed of the UAV and the radius of the circle are both kept constant as the guidance commands which are then used to generate centrifugal accelerations as the required flight commands.

Simulation Platform Setup

To study the differences between the nonreal-time single PC and real-time simulation, two 486 personal computers are used to form the real-time distributed parallel simulation platform as shown in figure 3^{11}. The controller PC-486(DX-33) contains the guidance software which takes in aircraft position and target position and estimates guidance commands. It takes in the measurements from the system dynamics and navigation system through analog to digital (A/D) interface card and sends out guidance commands through digital to analog (D/A) interface card. The system PC-486(DX2-66) receives the guidance signals from the guidance controller through A/D interface card, then derives control efforts and integrates the system dynamics model and the environment model. At the end of each integration step, the system states and the environment conditions are sent out to the controller PC through D/A interface card.

The two A/D cards are the Advantech® 818 with 8 A/D differential channels, where each channel has 12-bit resolution and ±10V range. Single channel data rate for 818 card is 1 kHz but down to 950 Hz for running 8 channels due to multiplexing of the channels. The two D/A cards are the Advantech® 726 with 6 D/A channels of 12-bit resolution and ±10V range. The data rate is about the same as the 818 card. To make the simulation as real as possible, the integration time-step of each PC has to be adjusted similar to the data transmission frequencies of the controller and the system sensors. Output D/A from the digital controller module has slower data transmission rate than the output D/A from the dynamics/environment module. Turbo C is used for software program coding.

Example

The F-16 model is used to simulate the performance of the integrated controller developed in this research. The aircraft initially is at (X,Y,Z)=(0,0,30000)ft with velocity V_x = 795ft/s. It is required to reach a way point at (20000,20000,300000)ft and then to fly a circle of radius 5000ft about the point (40000,40000,35000)ft. The parameters of the controller are given in the following table

τ_p	τ_q	τ_r	τ_x	τ_y	τ_z	λ_x	λ_y	λ_z
5	5	5	0.3	0.3	0.3	0.3	0.3	0.3
τ_v	K_v	K_p	K_q	K_r	K_x	K_y	K_z	K_η
0.1	5	1	2	1	20	20	20	10
ϵ_v	ϵ_p	ϵ_q	ϵ_r	ϵ_x	ϵ_y	ϵ_z	ζ	κ
2	0.5	0.01	0.5	10	10	10	0.012	0.02

The simulation results are shown in figures (3-7). From the figures, we notice that the flight trajectories from the nonreal-time single PC and the real-time platform are almost the same as shown in figure 3. However, In figure (4-7), we see some differences in the heading angles α, β and the control inputs η, δ_e,, δ_a, δ_r, $\delta_l ef$, especially in the locations when the UAV is changing its course. The example also demonstrates that the controller developed is stable during air flight and flight path changes. With the application of minimum norm method, the controller keeps the control inputs in reasonable bounds. To interpret the motion of the UAV, we notice from figure 7 that it is necessary to bank the aircraft first so that the component of lift force can provide the centrifugal force. As the aircraft is banked suddenly, it will induce sideslip motion due to gravity force at transition states.

Conclusions

To develop a nonlinear flight controller suitable for guidance purpose, aircraft dynamic model was separated first into a fast subsystem and a slow subsystem. The six-degree of freedom UAV is considered to have five control inputs. A constraint was imposed to satisfy the controller requirements which required the sideslip angle to be zero at all time. This condition implies that " bank to turn" is necessary. Due to modeling errors and uncontrollable internal dynamics arisen from the model separation and feedback linearization processes, the sliding mode control algorithm incorporating the dynamic extension method was used to ensure the robustness of the control system. The chattering phenomenon is avoided by including a saturation function to replace the sign function.

The flight controller consists a fast-mode controller and a slow-mode controller. Such configuration allows the guidance command to enter the slow-mode controller very easily and produce required attitude rate and throttle. The attitude rate is then entered the fast-mode controller to compute necessary deflections of the control surfaces. Numerical simulations for real-time and nonreal-time indicate stable flight and successful guidance maneuver can be achieved.

Acknowledgment

This work was partially supported by the National Science Council of R.O.C. under the project NSC85-2212-E-006-066.

References

1. S.J. Zaloga, 'Unmanned Aerial Vehicles', *Aviation Week & Space Technology*, January 8, 1996, pp.87.

2. W.D. Morse, 'Model Following Reconfigurable Flight Control System for the AFTI/F-16', *Journal of Guidance, Control and Dynamics*, Vol. 13 No.6, 1990. pp. 969-976.

3. S.A. Snell, D.F. Enns and W.L. Garrard Jr. 'Nonlinear Inversion Flight Control for a supermaneuverable Aircraft,' *Journal of Guidance, Control and Dynamics*, Vol. 15 No. 4, 1992. pp. 228-237

4. P.K.A. Menon, M.E. Badgett and R.A. Walker, 'Nonlinear Flight Test Trajectory Controllers for Aircraft,' *Journal of Guidance, Control and Dynamics*, Vol. 10 No. 1, 1987. pp. 67-72

5. Z. Xu and L.R. Hunt, 'Linear Dynamics Hidden By Input-Output Linearization for Square System,' *Proceedings of the 29th IEEE conference on Decision and Control*, 1990, pp. 786-791

6. A. Isidori, *Nonlinear Control Systems*, Springer- Verlad, 1989.

7. H. Elmali and N. Olgac, 'Robust Output Tracking Control of Nonlinear MIMO System via Sliding Mode Technique', *Automatica*, Vol. 28 No 1, 1992, pp. 145-151.

8. C. F. Chang, *Integrated Flight/Fire Control Via Nonlinear Transformation Approach*, Master Thesis, IAA, NCKU, Tainan, Taiwan, 1993.

9. D.L.Sheu, N.X.Vinh, and R.M. Howe, 'Application of Singular Perturbation Methods for three Dimensional Minimum Time Interception,' *Journal of Guidance, Control and Dynamics*, Vol. 28 No 1, 1991, pp. 360-367.

10. F.T. Cheng and D.E. Orin, 'Efficient Algorithms for Optimal Force Distribution - The Compact - Dual LP Method' *IEEE Trans Robotics Automat.*, Vol. 6 No 2, 1990, pp. 178-187.

11. Y.Y. Lin and S.S. Liu, 'PC-Based Distributed Parallel Processing for Real-Time Simulation of Spacecraft Attitude Control', *Proceedings of AIAA Modeling and Simulation Technology Conference*, New Orleans, LA, August 11-13, 1997, Paper No. 97-3675.

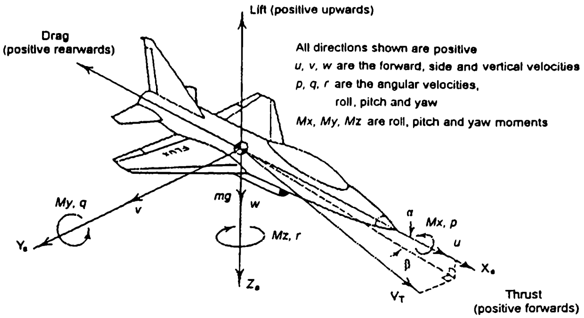

Lift (positive upwards)
Drag
(positive rearwards)
All directions shown are positive
u, v, w are the forward, side and vertical velocities
p, q, r are the angular velocities,
roll, pitch and yaw
Mx, My, Mz are roll, pitch and yaw moments
My, q
v
Y
mg
w
Mz, r
Z
α
Mx, p
u
X
β
V_T
Thrust
(positive forwards)

Figure 1 Parameters of UAV

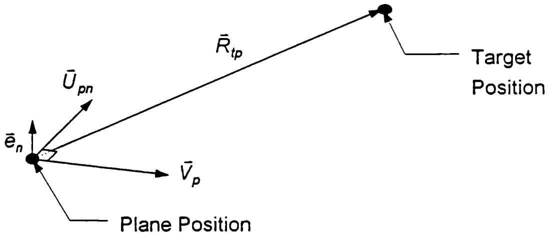


Figure 2 Projection guidance method

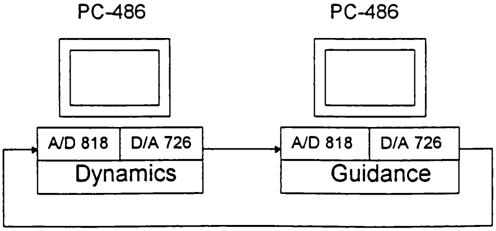


Figure 3 Real-time distributed parallel processing platform

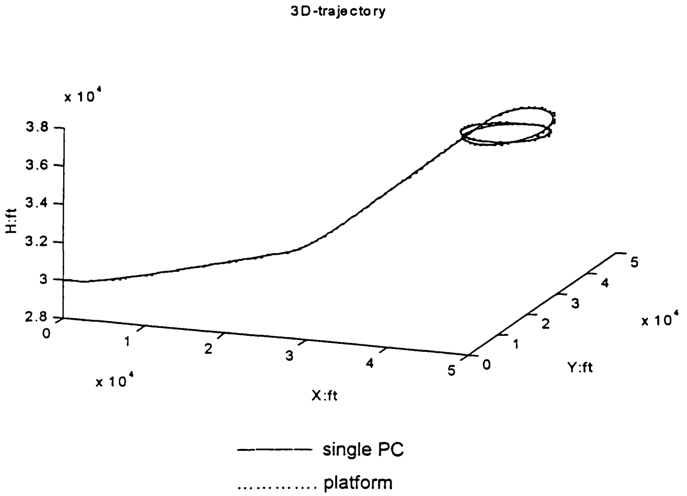


Figure 4 3D flight trajectory of UAV

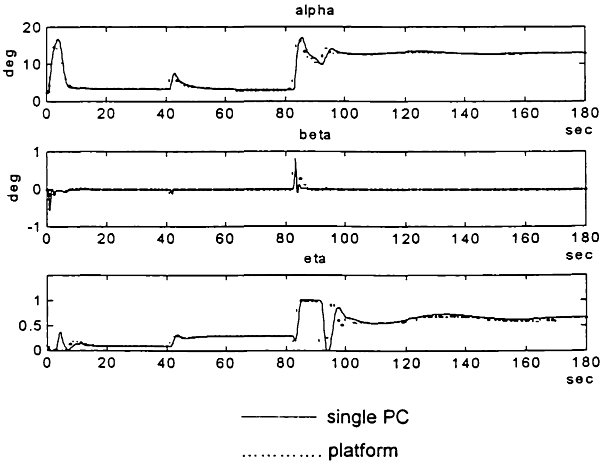


Figure 5 Angle of attack, sideslip angle and throttle

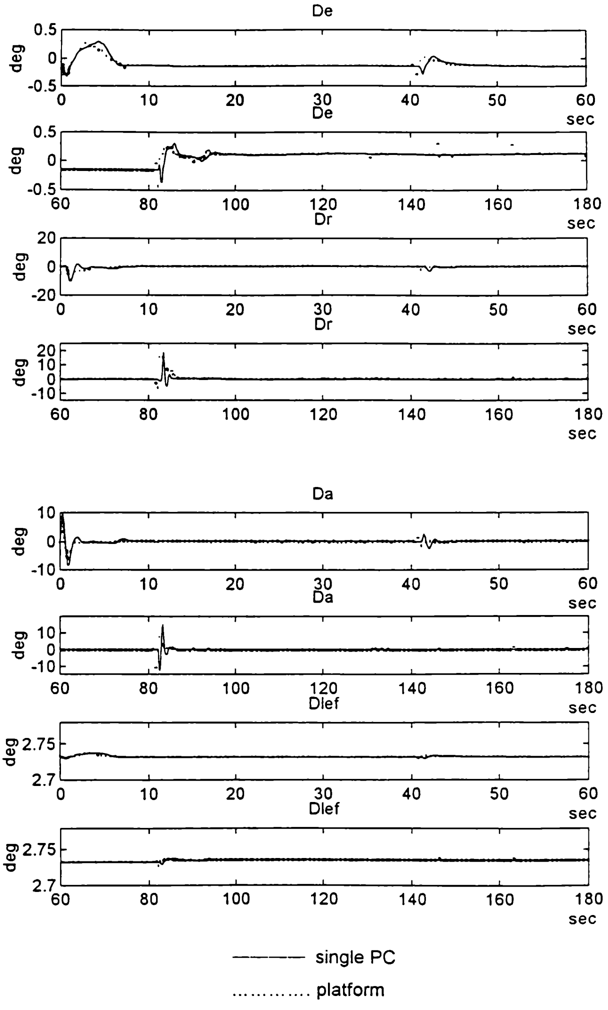


Figure 6 Control surfaces

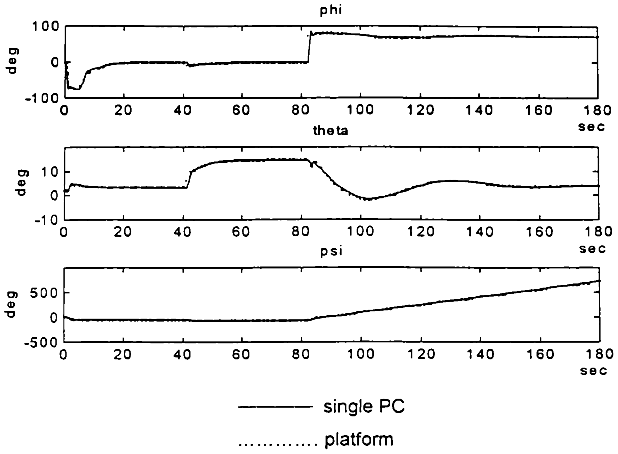


Figure 7 Attitude angles ϕ, θ, ψ

AIAA-98-4537

MODELING PILOT DECISION MAKING BY AN INFLUENCE DIAGRAM

Kai Virtanen*, Tuomas Raivio† and Raimo P. Hämäläinen‡
Systems Analysis Laboratory[§]
Helsinki University of Technology
P.O.Box 1100
FIN-02015 HUT
Finland

ABSTRACT

We apply an influence diagram to model pilot decision making in one versus one air combat. An influence diagram graphically describes the factors of a decision process under uncertainty and takes into account all available information to provide logical decisions. In the constructed model, the possible combat situations related to each maneuver alternative are associated with a probability and a utility. Influence diagram analysis produces probability distributions of the overall utility that are used in selecting the preferred maneuver. Sensitivity analysis determines the impacts of different factors on the outcome of the maneuvering decision. The effects of sensor information that will reduce the uncertainty of the model, are evaluated using Bayesian reasoning. The model can be utilized in the analysis of a single decision situation or as an automated decision making system that selects combat maneuvers in air combat simulators.

INTRODUCTION

In this paper a tool from decision analysis,[2,4,10,22] the influence diagram,[8] is applied to model the decision problems of a pilot during one versus one air combat. The constructed decision model offers a tool for analyzing different combat situations. On the other hand, it produces reasonable and logical combat decisions that can be utilized in simulation. Although influence diagrams have already been applied in mission planning,[6] pilot decision making has not been modeled using this tool.

Analysis of air combat tactics and technologies as well as pilot training are expensive tasks, and not all air combat situations can be analyzed in practice. For example, such a situation arises when a missile is trying to intercept an aircraft. Optimization, game theory and simulation are widely used in analyzing combat tactics and technologies. Optimal control theory can be utilized in studying optimal trajectories of a single aircraft.[15,21] Certain parts of air combat can be described as pursuit-evasion games[16], but if the roles of the players are not dictated, simulation is practically the only approach to analyze air combat mathematically. Thus different batch and real time piloted air combat simulators have been developed.[1,5,11,20] Batch simulators allow the study of combat tactics and aircraft performance in a controlled and repeatable environment. Real time piloted simulators enable tactical experimentation and training of human pilots in a realistic environment. Simulators of both type utilize computer guided aircraft.

One of the main components of a computer guided aircraft is a model that imitates pilot decision making. A decision model must be able to analyze an irregular flow of incoming data that describes a dynamically evolving environment. The available information is incomplete and the outcome of any particular action is at least partially uncertain. Furthermore, a decision model must involve competing objectives, such as the need to achieve a firing position and to simultaneously avoid the opponent's weapons.

Decision making models of the existing air combat simulators are knowledge based expert systems[5,12-20] that utilize computational techniques of artificial intelligence (AI), or heuristic value driven systems.[11] In these systems, decisions are made at discrete time instants. The future states of the opponent and the computer guided aircraft after a given planing horizon are first predicted and then evaluated by

[*] Researcher, e-mail: kai.virtanen@hut.fi

[†] Assistant Professor, e-mail: tuomas.raivio@hut.fi

[‡] Professor, Director of the Systems Analysis Laboratory, HUT, e-mail: raimo@hut.fi

[§] http://www.hut.fi/Units/Systems.Analysis/

assessing a score to each state. Finally, the decision alternative which leads to the highest score is executed. In the simplest form, a decision logic evaluates states by using predetermined geometry rules.[3] More advanced systems[5] apply a fixed set of questions defining the current combat situation and assess a value between zero and one to each question. The total value of each maneuver alternative is obtained by calculating the weighted sum of each single value. The weights characterize the relative importance of the single questions. States can also be scored by an explicit function that maps the combat situation into a value scale.[11] The function is additive and its weights are functions of time, as the importance of objectives change during an engagement.

Decision theoretical models and knowledge based expert systems are designed to model and improve human decision making. However, the approaches are based on rather different principles. Decision theory applies utility theory[10] and the axioms of probability. Expert systems involve applications of various logical and computational techniques. These systems have problems in modeling decision making under uncertainty since AI research has paid little attention to the modelling of human preferences and attitudes toward risk.[7] A realistic model for decision making under uncertainty should take into account the decision maker's preferences that in general have not been addressed by AI.

In an influence diagram model, utility functions[10] describe the preferences of the pilot. Tradeoffs between competing objectives are made by weighting utility functions[10] whereas in rule based systems the tradeoffs must be explicitly expressed. An influence diagram is flexible and it can be modified easier than a rule based system. Furthermore, a diagram can be constructed, validated and updated together with pilots because it is understood by individuals with little decision theoretical background. In AI, pilots can validate and analyze models only by simulating. Further differences are discussed in Ref. 7.

In the constructed model, competing objectives are measured in terms of the attributes, like the distance to the opponent, and velocity. The state of air combat defines the attribute values that are mapped into a commensurable utility scale using single attribute utility functions. Finally, the single utilities are aggregated to evaluate the different states of air combat. On the other hand, the model associates probabilities with states. The results of influence diagram analysis include probability distributions of utility for each decision alternative. The decision is based on a selected decision criterion. For example, we can choose the alternative that provides the highest expected utility. By using sensitivity analysis, important and critical factors of the given combat situation are determined. Value and effects of information gathering activities are analyzed using Bayesian reasoning, see, e.g., Ref. 4. These analyses will be introduced through the example decision situations of this paper.

INFLUENCE DIAGRAM

An influence diagram[8] (for more technical details, see Ref. 13) is a directed acyclic graph that graphically represents a decision process. It consists of arcs and nodes of three types: decision nodes, chance nodes and deterministic nodes, see Fig. 1. Arcs in a diagram show how each element interacts with each other. A decision node that includes different decision alternatives and can have numerical values associated with each alternative. A chance node represents an uncertain event or a random variable and has a numerical value and probability associated with each outcome. Deterministic quantities or variables are modeled by deterministic nodes whose value is either a constant or a function of its inputs.

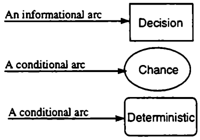


Figure 1. Nodes and arcs in an influence diagram. Decision nodes are represented by squares, chance nodes by ovals and deterministic nodes by squares with rounded corners.

The meaning of arcs in a diagram depends on their destination node. Conditional arcs leading into a chance or deterministic node represent probabilistic or functional dependence. They do not necessarily imply causality, although they often do. For example, an arc leading into a chance node indicates that an uncertain event has a different probability distribution and a different numerical value for each value and outcome of the preceding node. Informational arcs pointing to a decision node imply time precedence. They show which quantities are known to a decision maker before a decision is made.

A special deterministic utility node includes a utility function[10] that models the decision maker's preferences and, in practice, evaluates possible consequences of the decisions. Only one utility node can be designated, and it has no successor nodes.

States of the world are described using a set of attributes $x_1,\ldots,x_n$ that are related to competing objectives. For example, the pilot's objective "achieve a firing position" can be measured using the noncommensurable attributes "distance to the opponent" and "angle between the velocity vector of the pilot's aircraft and the line of sight". A single attribute utility function maps the value of an attribute into a utility scale such that the worst value of the attribute is mapped to zero and the best value to one, respectively. Finally, the single utilities are combined using an aggregating function.

The two common aggregating utility functions are additive and multiplicative. The form of the additive function is

$$u(x_1,\ldots,x_n)=\sum_{i=1}^{n} w_i u_i(x_i),$$

where x_i is the attribute i, u_i is a single attribute utility function and w_i are the positive weights that sum up to 1. The additive model is appropriate only when the decision maker's preferences are additively independent, see, e.g., Ref. 10. If additive indepence is not satisfied, a multiplicative form can be used for aggregation.

The weights of a multi-criteria decision model represent the importance of the attributes. In practice, the weights and the utility functions are extracted from the decision maker by using the methods for constructing single attribute utility functions and assessing weights, see, e.g., Refs. 9, 10. The decision maker's risk attitude defines the shape of the utility functions. A convex utility function indicates risk seeking and a concave risk averse behaviour.

In an influence diagram, probability distributions are updated based on Bayesian reasoning where the subjective probability interpretation, presented first by Ref. 17, is utilized. A subjective probability $P(\theta_i)$ is taken as representing the decision maker's degree of belief in the occurrence of an event θ_i based on the decision maker's current information. A decision maker can exploit several methods for assessing probabilities, see, e.g., Ref. 19.

Let us assume that only a finite number of states of the world are possible and label these states by $\theta_1,\ldots,\theta_n$. The decision maker's beliefs on the different states are given by the subjective prior probabilities $P(\theta_1),\ldots,P(\theta_n)$ such that $\sum_{i=1}^{n} P(\theta_i)=1$. These beliefs are utilized, if a decision must be made immediately. However, before the decision maker needs to make his decision, he observes that the event D has occurred. He now forms his posterior probabilities $P(\theta_1 \mid D),\ldots,P(\theta_n \mid D)$ and will take his decision considering these probabilities. The posterior probablity of belief is formed using Bayes' theorem:

$$P(\theta_k \mid D)=\frac{P(D \mid \theta_k)P(\theta_k)}{\sum_{j=1}^{n} P(D \mid \theta_j)P(\theta_j)},$$

where k=1,...,n, and the term $P(D \mid \theta_k)$ is called the likelihood probability. It means the probability that the event D is occured under the supposition that the state of the world is θ_k.

A complete influence diagram associates a probability and a utility with each possible consequence of the decision. An influence diagram analysis determines probability distributions of utility associated with each decision alternative. Based on these distributions, the best decision alternative is chosen. The decision criterion might be, e.g., maximum expected utility. However, expected utilities are not perfect indicators of what might happen since the risk of alternatives varies. Risks can be studied by looking at the distributions of the utility. For example, the decision alternative with the highest expected utility can also lead to a worse outcome with certain positive probability.

An influence diagram that includes only discrete probability distributions, can be converted into a decision tree[18] that is both a graphical representation of decisions, uncertainties and values and a framework for numerical evaluation of a model. Influence diagrams and decision trees have different advantages for modeling decisions and they complement each other.[13] If an influence diagram includes continuous probability distributions, it can be analyzed by using Monte Carlo simulation.

COMBAT DESCRIPTION

We consider one on one air combat in three dimensional space. There are two players, the decision maker and the opponent. The decision model is aimed at producing maneuvering and missile launching decisions for the decision maker. The aircraft of the decision maker is described using a three degrees of freedom point mass model. The state vector X is subject to the equations of motion

$$\dot{X}=f(X,n,\mu,u),$$

where the state vector X includes variables that refer to the x-range, the y-range, altitude, velocity, flight path angle, heading angle and mass. The normal acceleration of the aircraft is controlled with the loadfactor n and the tangential acceleration with the throttle setting u. The loadfactor can be directed with the bank angle μ. The gravitational acceleration and the aircraft mass are assumed to be constant. In the decision model, drag coefficients and maximum thrust force of a generic modern fighther aircraft are used. The Mach number and the air density are computed on the basis of the ISA standard atmosphere.

Values of the control variables are restricted by the constraints

$$n \in [n_{min}, n_{max}],\ u \in [0,1] \text{ and } \mu \in [-\pi, \pi].$$

The feasible region of stationary flight is described by the minimum altitude, the minimum velocity and maximum dynamic pressure constraints. For details of the model, see Ref. 21.

In the influence diagram model the continuous control variables n, u and μ are replaced with seven discrete control alternatives. Decisions are made at discrete time instants and the selected control is maintained during a fixed time interval Δt that is called the planning horizon or the lookahead time. The control alternatives are:

1: Maximal increase of the load factor
$n_{com} = n_{old} + n_{\Delta}\Delta t,\quad u_{com} = u_{old},\quad \mu_{com} = \mu_{old}$
2: Maximal decrease of the load factor
$n_{com} = n_{old} - n_{\Delta}\Delta t,\quad u_{com} = u_{old},\quad \mu_{com} = \mu_{old}$
3: Maximal increase of the bank angle
$n_{com} = n_{old},\quad u_{com} = u_{old},\quad \mu_{com} = \mu_{old} + \mu_{\Delta}\Delta t$
4: Maximal decrease of the bank angle
$n_{com} = n_{old},\quad u_{com} = u_{old},\quad \mu_{com} = \mu_{old} - \mu_{\Delta}\Delta t$
5: Maximal increase of the throttle setting
$n_{com} = n_{old},\quad u_{com} = u_{old} + u_{\Delta}\Delta t,\quad \mu_{com} = \mu_{old}$
6: Maximal decrease of the throttle setting
$n_{com} = n_{old},\quad u_{com} = u_{old} - u_{\Delta}\Delta t,\quad \mu_{com} = \mu_{old}$
7: The controls are held unchanged
$n_{com} = n_{old},\quad u_{com} = u_{old},\quad \mu_{com} = \mu_{old}$

Here the subscript *com* refers to the commanded values of the controls and the subscript *old* refers to the values of the controls at the decision instant. The control rates n_{Δ}, μ_{Δ} and u_{Δ} are fixed. In fact, this scheme introduces an additional order in the dynamics that approximates the pilot and the actuators.

The decision maker's predicted states after each control alternative are obtained by integrating the equations of motion with the control alternatives. A given maneuver alternative is ignored, if it leads to violation of the state or control constraints.

In our model, the relative geometry of an air combat situation is described using four attributes: deviation angle α_1, angle off α_2, distance between the players d and the angle between the players' velocity vectors β, see Fig. 2. Once one of the velocity vectors is known in the (x,y,h) frame, the description is unique.

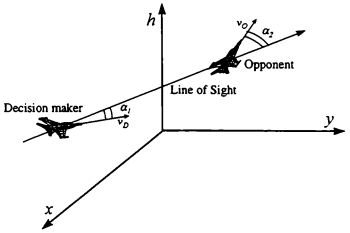


Figure 2. Relative geometry of air combat.

PILOT DECISION MODEL

The constructed influence diagram is shown in Fig. 3. It indicates that there are two decisions to be made in this decision process. The control decision is modeled by the *Maneuver* node and the use of the weapon by the *Launch Missile* node. The former node includes seven control alternatives described in the previous section. The latter node has two decision alternatives: *launch the missile* and *do not launch the missile.* The values of the predicted states related to each control alternative, are given in the deterministic node labeled *Predicted State*.

The deterministic node *Opponent's Predicted State* includes the exact values of the opponent's state. The opponent's state is described using x-range, y-range, altitude, flight path angle, heading angle and velocity. The opponent's and decision maker's states define the momentary relative geometry that is included in the *True Geometry* node. This deterministic node computes the combat description attributes that are introduced in the previous section.

The chance nodes *Sensor 1*, *Sensor 2* and *Sensor 3* model the decision maker's observations from the state of the opponent. It can be considered that the pilot can become aware of the opponent's state by seeing him visually, receiving radio communications from a battle manager or by detecting him with a radar. Each node produces observations from the opponent's x-range, y-range, altitude, flight path angle, heading angle and velocity. For the sensor j,

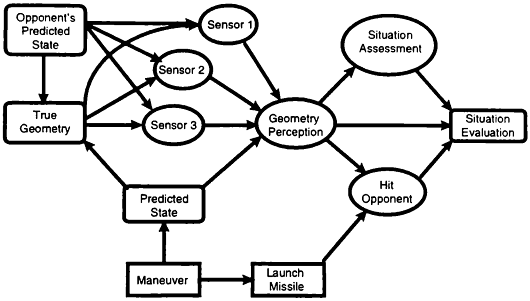


Figure 3: An influence diagram representing decision problems during air combat.

the observations $y_{i,j}=(y^1_{i,j},\ldots,y^{n_j}_{i,j})$ are generated from a normal distribution

$$P_j(x_i)=\frac{1}{\sqrt{2\pi\sigma^2_{o,i,j}}}e^{-\frac{1}{2\sigma^2_{o,i,j}}(x_i-\mu_{o,i,j})^2},$$

where $\mu_{o,i,j}$ is the expected value and $\sigma_{o,i,j}^2$ is the variance of the observation on the state variable x_i. The expected values and the variances depend on the values of the *Opponent's Predicted State* and *True Geometry* nodes. The sensors are assumed to be unbiased and thus the expected values are equal to the opponent's exact state. The variance can be considered as a measure for accuracy of the sensor. The decision maker's prior belief on the state of the opponent fixes the expected value $\mu_{pr,i,j}$ and the variance $\sigma_{pr,i,j}^2$ of the prior distributions. The posterior distributions are formed using Bayesian reasoning such that the number of observations n_j and the variance $\sigma_{o,i,j}^2$ are assumed to be known. In this case it can be shown, see, e.g., Ref. 4, that the posterior distribution $P_j(x_i|y_{i,j})$ is also a normal distribution whose expected value $\mu_{po,i,j}$ and variance $\sigma_{po,i,j}^2$ satisfy

$$\mu_{po,i,j}=\frac{\frac{1}{\sigma^2_{pr,i,j}}\mu_{pr,i,j}+\frac{n_j}{\sigma^2_{o,i,j}}\overline{y}_{i,j}}{\frac{1}{\sigma^2_{pr,i,j}}+\frac{n}{\sigma^2_{o,i,j}}},$$

$$\frac{1}{\sigma^2_{po,i,j}}=\frac{1}{\sigma^2_{pr,i,j}}+\frac{n_j}{\sigma^2_{o,i,j}},$$

respectively. Here $\overline{y}_{i,j}$ is the mean of the observations $y_{i,j}$. After the posterior distributions are formed, values of the state variables that are utilized in the *Geometry Perception* node, are generated from the posterior distributions.

The chance node *Geometry Perception* represents the decision maker's comprehension of the relative geometry of the current air combat situation. The node includes the same attributes as the *True Geometry* node, but due to the sensors the combat description attributes are uncertain.

The chance node *Situation Assessment* includes the outcomes that describe the situation of the current air combat from the decision maker's point of view. This node has the following four outcomes:

θ_1 = Neutral
θ_2 = Advantage
θ_3 = Disadvantage
θ_4 = Mutual disadvantage

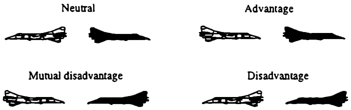


Figure 4: A sketch of the relative geometry of combat associated with each outcome of the *Situation Assessment* node. □ decision maker, ■ opponent.

The relative geometry of combat that is associated with each outcome of the *Situation Assessment* node, is sketched in Fig. 4. The decision maker's prior beliefs on the situation are given by $P(\theta_1)$, $P(\theta_2)$, $P(\theta_3)$, and $P(\theta_4)$ such that $\sum_{i=1}^{4}P(\theta_i)=1$. This distribution characterizes the nature of the air combat at a particular time. Suppose now that the decision maker does not have to make the decision immediately, but he can update his or her beliefs on the situation based on the observed values of α_1, α_2 and d. The decision maker's posterior belief on the current air combat situation is again calculated by Bayes' theorem:

$$P(\theta_i|\alpha_1,\alpha_2,d)=\frac{P(\theta_i)P(\alpha_1,\alpha_2,d|\theta_i)}{P(\alpha_1,\alpha_2,d)},\qquad i=1,\ldots,4.$$

Here α_1, α_2 and d are assumed to be independent random variables and thus

$$P(\alpha_1,\alpha_2,d|\theta_i)=P(\alpha_1|\theta_i)P(\alpha_2|\theta_i)P(d|\theta_i)\cdot$$

The likelihood probabilities $P(\alpha_1|\theta_i)$, $P(\alpha_2|\theta_i)$ and $P(d|\theta_i)$ can be formed using a pilot's experience in air combat. The total probability $P(\alpha_1,\alpha_2,d)$ is

$$P(\alpha_1,\alpha_2,d)=\sum_{i=1}^{4}P(\alpha_1,\alpha_2,d|\theta_i)P(\theta_i)\cdot$$

The probability that the launched missile will hit the opponent is modeled by the chance node labeled *Hit Opponent*. The outcomes are

ϕ_1 = The missile hits
ϕ_2 = The missile does not hit

Prior probability distributions of the uncertain outcomes $P(\phi_1)$ and $P(\phi_2)$ must again be specified in advance. Posterior distributions are calculated using Bayesian reasoning and they are based on outcomes of the chance node labeled *Geometry Perception*.

The goals of the decision maker depend on the current air combat situation. Thus, each outcome of the *Situation Assessment* node θ_i, i=1,...,4, and the *Hit Opponent* node ϕ_j, j=1,...,2, leads to a different preference order. Each combination of the outcomes of the nodes is connected to different utility function in the *Situation evaluation* node. The utility functions related to the outcome "*The missile hits*" are

$$u_i(\alpha_1,\alpha_2,\beta,d,L,v)=w_{\alpha_1}^{i}u_{\alpha_1}^{i}(\alpha_1)+w_{\alpha_2}^{i}u_{\alpha_2}^{i}(\alpha_2)+$$
$$w_{\beta}^{i}u_{\beta}^{i}(\beta)+w_{d}^{i}u_{d}^{i}(d)+w_{v}^{i}u_{v}^{i}(v)+w_{L}^{i}L,\quad i=1,\ldots,4,$$

and the functions related to the outcome "*The missile does not hit*" are

$$u_i(\alpha_1,\alpha_2,\beta,d,L,v)=w_{\alpha_1}^{i}u_{\alpha_1}^{i}(\alpha_1)+w_{\alpha_2}^{i}u_{\alpha_2}^{i}(\alpha_2)+$$
$$w_{\beta}^{i}u_{\beta}^{i}(\beta)+w_{d}^{i}u_{d}^{i}(d)+w_{v}^{i}u_{v}^{i}(v)+w_{L}^{i}(1-L),\quad i=5,\ldots,8,$$

respectively. Here v is the velocity of the decision maker and L is a binary variable whose value is 1 if the missile is launched and 0 if the missile is not launched. The aggregated utility functions u_i map the current air combat situation into a utility scale. Single attribute utility functions $u_{\cdot}^{i}$ and weights $w_{\cdot}^{i}$ describe the preferences of the decision maker. Here, the utility functions are selected somewhat freely to pick the essential characters of critical combat situations. For example, the utility functions of the deviation angle related to each outcome of the *Situation Assessment* node are shown in Fig. 5.

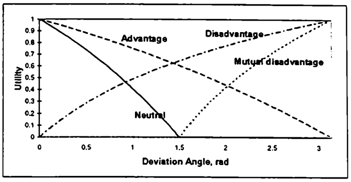


Figure 5: The utility functions of deviation angle related to the outcomes of the *Situation Assessment* node.

USE OF THE MODEL

ANALYZING AN AIR COMBAT SITUATION

We construct and solve the model by using the PrecisionTree software,[14] that is the add-in package for Microsoft Excel. PrecisionTree provides all the necessary tools for setting up and analyzing a decision model on a spreadsheet.

We next analyze an example air combat situation. At the decision instant the state of the decision maker is

$x = 0$ m, $y = 4200$ m, $h = 8000$ m, $v = 300$ m/s,
$\gamma = 0.2$ rad, $\chi = 0$ rad and $m = 10000$ kg.

The values of the control variables are $\mu = 0.5$ rad, $n = 1.5$ and $u = 0.5$, and the control rates are $\mu_\Delta = 1$ rad/s, $n_\Delta = 1$ 1/s and $u_\Delta = 0.5$ 1/s.

The exact state of the opponent is $x = 7000$ m, $y = 7000$ m, $h = 10000$ m, $v = 300$ m/s, $\gamma = 0$ rad and $\chi = 3.14$ rad, i.e., the opponent is approaching the decision maker in a higher altitude from top left.

Let us assume that the decision maker's posterior belief on the opponent's state is same as the exact state at the decision instant. Therefore, the expected values of the posterior distributions that model the decision maker's belief on the opponent's state, are equal to the exact values given above and the variances are very small. The prior distributions of the *Situation Assessment* and *Hit Opponent* nodes are assumed to be as follows:

$$P(\theta_1)=0.225,\ \ P(\theta_2)=0.175,\ \ P(\theta_3)=0.275,$$
$$P(\theta_4)=0.325\ \text{and}\ P(\phi_1)=0.4,\ \ P(\phi_2)=0.6.$$

The decision maker's states related to each control alternative, are predicted using the planing horizon of 10 seconds.

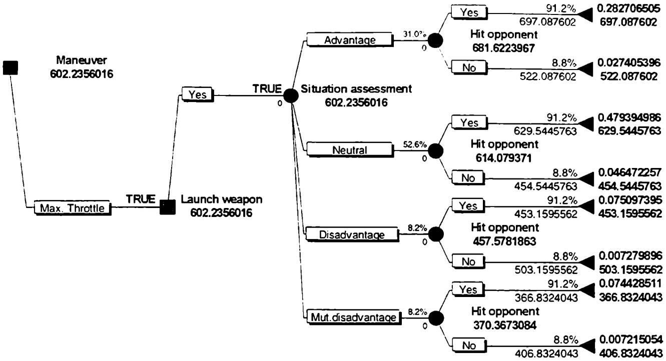


Figure 6. The policy suggestion for the air combat situation.

The influence diagram is solved by converting it into a decision tree and by solving the tree. In the example air combat situation, the maximum expected utility is obtained when the maneuvering decision is "maximal increase of throttle setting" and the missile is launched. This path of the decision tree is shown in Fig. 6.

Although the decision alternative with the highest expected utility is executed, there is a probability that the coming combat situation will be worse or better than the decision maker assumes. Probability distributions of utility graphically display uncertainty of decisions. In the example, the selected decision alternatives lead to the distribution shown in Fig. 7.

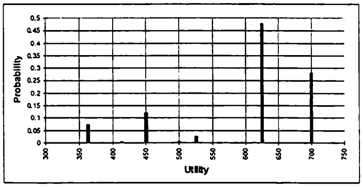


Figure 7. The probability distribution of utility for the decisions that maximize the expected utility.

Probability distributions of utility can also be constructed for other decision alternatives. The cumulative distributions for each maneuver alternative are shown in Fig. 8. By looking at this picture, dominated and dominating decision alternatives can be identified. For example, we can see that the distribution related to the alternative "maximal increase of the throttle setting" lies to the right of the distribution of "maximal increase of the load factor" and thus we can conclude that the former alternative leads to the better outcome with a higher probability than the latter alternative.

An expected utility is not the only possible measure of a probability distribution. Standard deviation is a measure of how widely the values are dispersed in a distribution and thus it is an indication of the risk. Minimal and maximal possible utilities indicate the worst and the best possible outcome that can occur. These quantities for each maneuver alternative in the example combat situation are shown in Table 1.

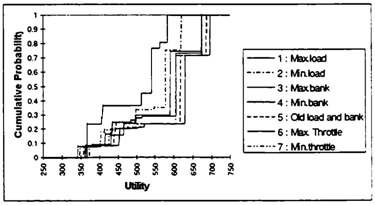


Figure 8. The cumulative probability distributions for the maneuver alternatives.

Maneuver Statistics	Max. load	Min. load	Max. bank	Min. bank	Old controls	Max. throttle	Min. throttle
Mean	558	537	574	489	590	602	578
Minimum	341	373	362	366	357	367	347
Maximum	673	620	674	584	686	697	674
Variance	105	85	98	85	101	101	101

Table 1: The measures that characterize probability distributions of utility.

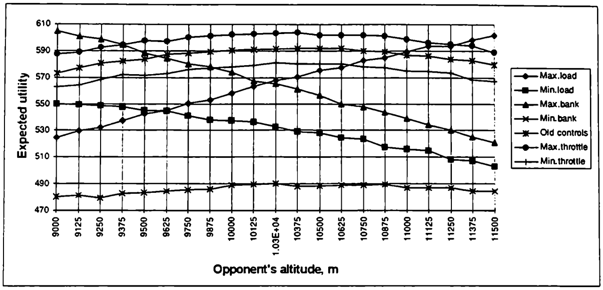


Figure 9. A sensitivity analysis with respect to the opponent's altitude.

Based only on the expected utilities, the control alternative "maximal increase of throttle setting" seems to produce the best outcome. The decision alternative can also be selected on the basis of maximin or maximax criteria.[2] Maximin is the most pessimistic criterion. First, for each alternative the worst possible (minimal) value of utility is identified. Then the alternative whose worst possible utility is highest, is chosen. Extremely optimistic decision maker looks at the best that can happen and then the maximax criterion is used. The objective is to find a decision alternative that gives the largest possible utility overall.

SENSITIVITY ANALYSIS

Sensitivity analysis determines the impacts of variables on decisions and outcomes. Thus, the most important factors in the given decision situation can be found out. There are many ways to run a sensitivity analysis on the pilot decision model. None of these methods are better than the others, but each method gives different information.

One way sensitivity analysis studies the effect of a single variable on expected utilities. As an example sensitivity analysis, the impact of the opponent's altitude on the maneuvering decision is studied. The previous air combat situation is referred as the base case, and all the subsequent results are compared with the outcome of this case.

The minimum value of the opponent's altitude is assumed to be 9000 m and the maximum value 11500 m, respectively. During the sensitivity analysis, 20 equally spaced values across the range of altitude are tested. The expected utilities related to each maneuver alternative are shown in Fig. 9. According to this figure, the best maneuver decision is the same as in the base case, when the opponent's altitude is approximately between 9400 m and 11300 m. When the altitude decreases below 9400 m, the control alternative "maximal increase of the bank angle" becomes the best decision. If the altitude is above 11300 m, the best action is to increase the load factor.

One way sensitivity analysis can also be made with respect to other opponent's state variables. For example, the effects of the opponent's altitude, y-range and x-range are studied. The y-range and x-range vary from 6000 m to 8000 m and altitude from 9000 m to 11000 m, respectively. The range of each variable is equal. The results of this analysis are shown in Fig. 10.

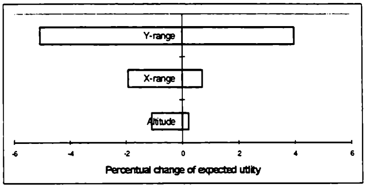


Figure 10. The impacts of the opponent's y-range, x-range and altitude on the expected utility.

The horizontal axis of the figure is plotted in the units of percentual change of the expected utility from the base case. The bars of the picture represent the percentual change of the expected utility when the specified opponent's state variable is varied from one end to the other, keeping all other state variables at their base values. The vertical line indicates the expected utility that is obtained in the base case. Now we can see that the y-range is the most important factor from among three variables that are analyzed. The expected utility is insensitive to the opponent's altitude, so in the further analysis, we can leave the altitude at its base value. In this way, less important factors of the decision situation can be identified and singled out.

Previous analysis gives significant information about the decision situation, although it is limited to what happens when only one variable varies at a time. The impact of two variables on a decision model can be studied using a two way sensitivity analysis where two variables vary simultaneously and at the same time all other variables are kept at their base case values. Typically, the two most critical variables are studied. The results of a two way sensitivity analysis show regions of the variables where different decisions are best. As an example, the change in the expected utility with respect to the opponent's x-range and y-range is shown in Fig. 11.

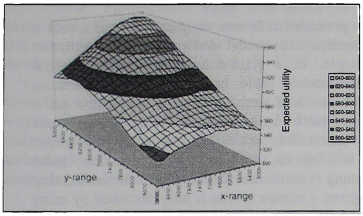


Figure 11. The two way sensitivity analysis with respect to the opponent's x-range and y-range which vary from 5000 m to 9000 m.

EFFECT AND VALUE OF SENSOR INFORMATION

In the constructed influence diagram, the *Sensor 1*, *Sensor 2* and *Sensor 3* nodes model the decision maker's information gathering activities. In this section, we analyze effects and value of sensor information by using the diagram.

Since the sensor nodes include continuous probability distributions, impact of new information can be studied using Monte Carlo simulation. Another approach would be to discretize continuous distributions and to solve a resulting decision tree with different information.[18] In Monte Carlo simulation, values of uncertain quantities are generated each according to its own probability distribution and once all the values have been determined, the expected utility of each decision alternative is calculated. This procedure is repeated many times and the results are recorded. At the end, it is possible to calculate the distributions of expected utility and to examine descriptive statistics of distribution such as the mean, the standard deviation and the maximum and minimum of the expected utility.

In this example, maneuver alternatives are put in order with respect to the mean of expected utility. The difference between means of expected utility is used as a measure of information value.

In practice, the decision maker does not know the opponent's exact state. Let us assume that his prior belief on the opponent's state is same as the base case state. Expected values of the distributions that model sensor observations, are equal to the opponent's exact state. Accuracy of the sensors is different such that the sensor 3 is the most accurate and the sensor 1 is the most inaccurate. The variance of the distribution that models the sensor 3, is the smallest and the variance related to the sensor 1 is the largest, respectively. The opponent's true state is now

$$x = 5000 \text{ m}, y = 5000 \text{ m}, h = 8000 \text{ m},$$
$$v = 300 \text{ m/s}, \gamma = 0 \text{ rad and } \chi = 1.5 \text{ rad}.$$

First, the decision maker does not receive extra information and thus the executed decision alternative is chosen based on his or her prior belief on the opponent's state. The cumulative probability distributions of the expected utility for each maneuver alternative are shown in Fig. 12. The maneuver alternative "maximal increase of the throttle setting" produces the highest mean of expected utility (644 utility points). Furthermore, this maneuver alternative ensures the highest expected utility in the worst and best possible case.

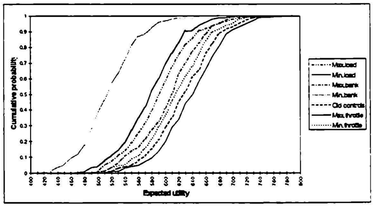


Figure 12: The cumulative probability distributions of expected utility when the decision is made based on prior information.

If the decision maker had access to perfect state information, the maneuver alternative "maximal decrease of the load factor" would lead to the highest mean of expected utility (745). We can see that prior information leads to the different maneuver alternative (maximal increase of the throttle setting). The mean of expected utility with prior information was 644 utility points and thus the value of perfect information, the difference between the means of expected utility, is 745-644=101. So, if the decision maker knows the opponent's state perfectly, he could gain 101 utility points more than if he only has prior information about the opponent's state.

Next, the influence diagram is solved six times using different combinations of the sensors and numbers of observations, see Table 2. For example, in the fourth case, the decision maker receives ten observations from the opponent by using sensor 1. The maximum mean of expected utility (716) is obtained with the maneuver alternative "maximal increase of bank angle". The value of information is 716-644=72. The executed maneuver alternative is not the same as in the case of perfect information, but it leads to the better outcome compared with the maneuver alternative that is obtained with prior information.

	Number of observations					
Case	Sensor 1	Sensor 2	Sensor 3	Alternative	Mean E(u)	Value
Prior	-	-	-	Max. throttle	644	-
1	1	0	0	Max. throttle	679	35
2	1	1	0	Max. throttle	695	51
3	1	1	1	Max. bank	715	71
4	10	0	0	Max. bank	716	72
5	10	10	0	Min. load	732	88
6	10	10	10	Min. load	739	95
Perfect	-	-	-	Min. load	745	101

Table 2: Executed maneuver alternatives, maximum means of expected utility and value of extra information related to different sensor information.

The summary of the results that are obtained with different information from the opponent's state, is shown in Table 2. We can see that the value of information approaches the value of perfect information, when the decision maker makes more observations. Furthermore, the changes in the preferred maneuver can be obtained. For example, for the first time the maneuvering decision is the same as the one obtained with perfect information, when ten observations are made using sensors 1 and 2.

By analyzing the pilot decision model with different information, impacts of the sensors on the outcome of pilot's decision situation can be studied. In this way, effects of sensor accuracy can be studied and the most important as well as critical sensors can be located.

MODEL ASSESSMENT

In addition to the examples presented in this paper, the influence diagram can also be utilized in other tasks. Sensitivity analysis can be used to assess the impact of model parameters. For example, effects of maximum thrust force or drag coefficients on the outcome of the pilot's decision situations can be studied. Furthermore, the consequences of the pilot's preferences that are results of tactics, training and doctrine, can be analyzed by varying the weights and changing the shape of the utility functions.

The diagram could also be used as a guidance system that selects combat maneuvers in air combat simulation. At the beginning of the simulation, the prior probability distributions of the model can be assumed to be uniform. During the simulation, the probabilities can be updated such that the prior distributions at the current decision instant can be associated with the posterior probabilities of the previous decision instant.

The presented influence diagram is by no means a comprehensive model but rather an illustrative example. In a refined model, several potential improvements could be investigated. Preferences and behavior of human pilots can be captured into a refined model by constructing utility functions and probability distributions in co-operation with pilots. In addition to the current sensor models, radar warning systems should be modeled. In the existing diagram, a missile system is only described by using an ad hoc probability of hit. A modification to a diagram should include more realistic missile systems that consist of guidance laws and aerodynamic models. Probabilities of hit produced by real weapon systems could also be utilized. Furthermore, models of guns should possibly be considered.

In the presented diagram, the opponent's state is modeled using deterministic variables. To achieve more realism, the opponent's future state can be predicted by making presumptions over the opponent's behavior. For example, a diagram could include chance nodes representing maneuvers that are rational for the opponent. Furthermore, the players' future states should be predicted further than one planing horizon ahead. In practice, a model of this type can be considered as a sequence of influence diagrams that produces better long time interval decision sequences. An influence diagram could also be extended into situations where there are several opponents and friendly aircraft.

CONCLUSION

Because of the complex and transient nature of air combat, a pilot faces complicated decision making problems and thus it is difficult to choose actions that lead to the best possible outcome. Therefore, a model that imitates pilot decision making, must have the capability to evaluate decision alternatives that have multiple conflicting objectives and whose outcomes are not known for certain. A model must also utilize new information that may reduce uncertainty. These features are taken into account in the presented influence diagram model.

The examples of this paper show how single decision situations of a pilot can be analyzed. The result of the analysis is the overall probability distributions of utility for each maneuver alternative. Utility represents the pilot's opinion on the combat situation. Based on utility distributions, a maneuver is suggested. In this way, logical decisions in the light of all the available information are obtained. Sensitivity analysis is carried out to determine which factors in the given decision situation are important and critical. Furthermore, value and effect of observations from the opponent can be analyzed. In this way, one can produce useful instructions for the development and selection of sensors.

Overall, an influence diagram analysis provides a way to analyze the pilot's preferences as well as to compare the performance of different aircraft and technologies. This new approach holds promise for improving the understanding of air combat.

REFERENCES

[1] Bent, N.E., "The Helicopter Air-To-Air Value Driven Engagement Model (HAVDEM)," *Proceedings of American Institute of Aeronautics and Astronautics Conference*, ICAS-94-8.6.1, 1994, pp. 2181-2189.

[2] Bunn, D., *Applied Decision Analysis*, McGraw-Hill, New York, 1984.

[3] Burgin, G.H., and Sidor, L.B., "Rule-Based Air Combat Simulation," NASA, CR-4160, 1988.

[4] Clemen, R.T., *Making Hard Decisions, An Introduction to Decision Analysis*, 2nd ed., Duxbury Press, Belmont, CA, 1996.

[5] Goodrich, K. H., and McManus J. W., "Development of A Tactical Guidance Research and Evaluation System (TiGRES)," *Proceedings of American Institute of Aeronautics and Astronautics Conference*, Paper 89-3312, Aug. 1989.

[6] Griggs, B.J., Parnell, G.S., and Lehmkuhl, J.L., "An Air Mission Planning Algorithm Using Decision Analysis and Mixed Integer Programming," *Operations Research*, Vol. 45, No. 5, Sep./Oct. 1997, pp. 662-676.

[7] Henrion, M., Breese, J. S., and Hortvitz, E. J., "Decision Analysis and Expert Systems," *AI Magazine 12*, No. 4, 1991, pp. 64-91.

[8] Howard, R.A., and Matheson, J.E., "Influence Diagrams," In Howard, R.A., and Matheson, J.E., (eds.), *The Principles and Applications of Decision Analysis*, Vol. 2, Palo Alto, CA: Strategic Decision Group, 1984, pp. 719-762.

[9] Hämäläinen, R.P., and Lauri, H., "HIPRE 3+ User's guide," Systems Analysis Laboratory, Helsinki University of Technology, 1992.

[10] Keeney, R., and Raiffa, H., *Decision with Multiple Objectives*, Wiley, New York, 1976.

[11] Lazarus, E., "The Application of Value-Driven Decison-Making in Air Combat Simulation," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, Florida, USA, October 12-15, 1997, pp. 2302-2307.

[12] McManus, J. W., and Goodrich, K. H., "Application of Artificial Intelligence (AI) Programming Techniques to Tactical Guidance For Fighter Aircraft," *Proceedings of AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 89-3525, 1989, pp. 851-858.

[13] Oliver, R.M., and Smith, J.Q., *Influence Diagrams, Belief Nets and Decision Analysis*, Wiley, New York, 1989.

[14] *Precision Tree, User's Guide*, Palisade Corporation, 1996.

[15] Raivio, T., Ehtamo, H., and Hämäläinen, R.P., "Aircraft trajectory optimization using nonlinear programming," In: *System Modeling and Optimization*, Dolezal, J., and Fidler, J., (eds.), Chapman & Hall, 1996, pp. 435-441.

[16] Raivio, T., Ehtamo, H., and Hämäläinen, R.P., "A decomposition method for a class of pursuit-evasion games," *Proceedings of the 7th International Symposium on Differential Games and Applications*, Vol. 2, Kanagawa, Japan, December 16-18, 1996, pp. 784-795.

[17] Savage, L.J., *The foundations of Statistics*, Wiley, New York, 1954.

[18] Shacter, R., "Evaluating Influence Diagrams," *Operations Research*, 34, 1986, pp. 871-882.

[19] Spetzler, C.S., and Stael von Holstein C.A., "Probability Encoding in Decision Analysis," *Management Science*, 22, 1974, pp. 340-352.

[20] Stehlin P., Hallkvist I., and Dahlstrand H., "Models for Air Combat Simulation," *Proceedings of American Institute of Aeronautics and Astronautics Conference*, ICAS-94-8.6.2, 1994, pp. 2190-2196.

[21] Virtanen, K., Ehtamo, H., Raivio, T., and Hämäläinen, R.P., "VIATO - Visual Interactive Aircraft Trajectory Optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C, Applications and Reviews*, to appear.

[22] von Winterfeldt, D., and Edwards, W., *Decision Analysis and Behavioral Research*, Cambridge University Press, Cambridge, 1986.

AIAA-98-4538

SIMULATION USE FOR TRAINING LARGE TACTICAL TEAMS

James Gualtieri, Ph. D., Maureen Bergondy, Randall Oser, and
Naval Air Warfare Center Training Systems Division*
Orlando, FL
Jennifer Fowlkes, Ph. D.
Summit Technologies, Inc.*
Orlando, FL

Abstract

The opportunity for naval aviators to develop coordination skills and acquire the knowledge necessary for integration in large tactical teams (e.g., Navy carrier air wings) during training is currently limited. Simulations and simulators could provide greater opportunity for the training of these critical skills. This paper will describe an investigation performed with a goal of defining potential training needs and strategies for large tactical team training. Two sources of data were analyzed; a database of critical incidents and questionnaires collected during a large tactical team training setting. The results of the analyses are presented and preliminary strategies for conducting large tactical team training are forwarded. Finally, the implications of the findings for future research in this area are discussed.

Introduction

The trend in modern warfare is toward an increasing reliance on distributed systems. Concepts such as "network-centric" warfare suggest that future missions will be accomplished by complex teams who are physically dispersed. Research conducted to address technical issues related to this problem has advanced considerably in the last decade (e.g., construction of distributed interactive simulations, high level architectures, and large scale networks). However, the study of human performance in dispersed teams and distributed training methodologies has not kept pace. Therefore, research is needed to investigate how best to exploit technology in support of distributed training and to derive principles that will provide guidance in this crucial area.

Given the criticality of the missions these teams must execute, it is important to develop training systems and programs to support coordination and problem solving in these environments. To respond to this training need, it is necessary to understand: (1) What knowledge, skills, and attitudes need to be learned and trained to facilitate expert problem solving teams? and, (2) How to best utilize simulators and simulations for establishing an environment in which learning can take place? While little work has been done to answer these questions for large tactical teams, recent findings from research in the areas of Naturalistic Decision Making (NDM)[1], Critical Teamwork Behaviors (CTB)[2], and Shared Mental Models (SMM)[3] provide a basis for beginning to answer these two questions.

The current paper presents an effort to establish the training needs for one type of distributed large tactical team, namely a Navy Carrier Air Wing. The paper will specifically: (1) provide a brief description of a carrier air wing, (2) forward a conceptually-derived framework used to examine training within the carrier air wing environment, (3) provide an overview of a data collection effort within a carrier air wing training environment, (4) present the results from analyses performed on the data, (5) describe the implications of the results for the use of simulators and simulations for training, (6) discuss the application of these findings for other types of large tactical teams, and (7) forward future requirements for additional research and development.

Navy Carrier Air Wing Teams

Navy carrier air wings are an example of one type of large distributed tactical team that must integrate a wide range of different types of assets for effective performance. Unfortunately, most existing training focuses on the competencies at the individual and aircrew level. Within naval aviation there exists a requirement to take aviators from the Fleet Replacement Squadrons (FRS) and provide them with training that will allow them to quickly acquire the additional skills and knowledge necessary to coordinate with other platforms and to become integrated into a carrier air wing. With the ever-increasing costs associated with training on instrumented ranges, it is necessary to ensure that the greatest benefit is achieved from every flight hour.

The strike missions that today's naval aviators must execute require the integration of a large tactical

The views expressed in this article are those of the authors and do nto represent the official positions of the organizations of the organizations with which they are affiliated.

team. A given mission may involve over 100 aviators flying in multiple platforms to attack a single target. Successful completion of the mission requires that a variety of subteams must be orchestrated to fulfill specific mission functions, including: strike, command and control (C2), suppression of enemy air defense (SEAD), and fighter air interdiction. In order to accomplish these mission functions, the subteams must have the ability to practice the requisite skills and learn essential knowledge. For example, the subteams must have the knowledge and skills to coordinate timelines, properly position aircraft, communicate effectively, and conduct maneuvers that require the adaptive integration of diverse and dispersed platforms.

The opportunity for naval aviators to develop these integration skills and acquire this knowledge during training is currently limited. Simulations and simulators could provide greater opportunity for the training of these critical skills. However, because of the size of the team and unique coordination demands, the evaluation strategies and metrics used for assessing training in single platform aircraft simulators may not be applicable at a Navy air wing level. The use of distributed interactive simulations (DIS) technology might afford a means by which large tactical teams could be trained. To ensure that the training in the simulated environment transfers to operational settings, appropriate tools, methods, evaluation strategies and metrics need to be identified. These mechanisms need to be derived from an understanding of the overall coordination processes found in Navy air wings. The following section presents a framework of the coordination processes derived from data recently collected from several Navy air wings during pre-deployment training.

Framework for Air Wing Coordination

Prior to air wing missions, there are a considerable number of planning and preparation tasks which must often be performed within a limited time frame. The first step in the process involves a small team with representatives from each of the air wing elements who must coordinate with a mission leader to develop the concept of operations, an overall mission plan, specific timelines, contingency plans, and detailed element plans. This information is then presented in an "overall" mission briefing to all members of the strike package. The overall brief is intended to provide the tactical picture, element coordination requirements, roles and responsibilities, and the tactical picture of the entire mission execution. The large team then breaks down into its subteam elements for function-specific briefs. These element briefs are intended to provide detailed information on subteam coordination and tactical requirements.

Because there is very little opportunity for strategizing during mission execution, implicit coordination among the strike package elements is essential (cf. [4]). The team members must have a clear understanding of task demands, team coordination requirements, team member responsibilities, contingency plans, and the environmental/mission cues that signal the need for the alternate plan. We contend that the plan and the brief provide the mechanisms to set up effective team coordination by building mental models of the tasks (missions), interdependencies, and contingencies. Effective planning and briefing are expected to improve implicit coordination and thus, mission performance of the large distributed team (Figure 1).

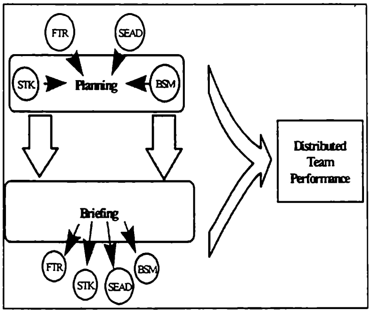


Figure 1. Framework for Carrier Air Wing Coordination

While considerable advancements have been made in the area of team training research over the past decade, many research and training issues have not specifically addressed the challenges confronting large distributed tactical teams. For example, with large tactical aviation teams, it is probably of greater importance than with other teams to build the big picture, or "mission concept," prior to mission execution. Because of the sheer size of the team, and because communications are necessarily restricted during mission execution, it is difficult to regain situation awareness once it is degraded and opportunities for adaptability are limited. Thus, there may be greater emphasis on pre-mission planning and briefing.

Although little research exists at the air wing level, the results within the other team training and performance domains may be leveraged for this domain. For example, the concept of mental models has been increasingly invoked as a useful mechanism to explain the performance of teams. In addition, there has been progress in the development of concepts for the training of distributed teams based on a constructivist approach.[5] Both of these areas have produced concepts and guidelines that may be applied to the training of large tactical teams. The specific application of concepts and guidelines for training must be based upon what constitutes effective learning and how effective learning can be realized using existing and emerging technology.

Effective Learning

Learning is both a cognitive and behavioral process. The missions and tasks to be trained in a distributed team are often very complex and situationally dependent. Effective performance requires competencies in a wide range of skills and abilities including: strategic planning, situation assessment, decision making, and resource management. These processes are compromised by the limited interface opportunities of distributed teams. Often, performance is further complicated because there may be few procedures and more than one approach to successfully accomplishing a mission. Because of these issues, training needs to focus on developing and assessing the team members' skills (i.e., behaviors) required to perform in a distributed environment.

Effective learning also needs to address the cognitive differences between new and veteran team members. One area in which differences would be expected between these two groups is the manner in which they structure knowledge. These knowledge structures have been referred to as mental models. Given that the mental models of veteran and novice team members are predicted to be vastly different, the question is not just how can we provide the novice with the additional knowledge necessary to interact effectively with the veteran, but how do we adapt the structure of the novice's knowledge to help with the acquisition and integration of that additional knowledge. Therefore, it is necessary to structure the novice's mental model in such way that the novice will continue to learn as they transition from the schoolhouse (e.g., FRS) to an operational unit (e.g., an air wing).

Simulations and Training

Recent advancements in technology (e.g., modeling, simulation, networking) has resulted in the ability to create shared virtual environments.[6] While these advancements can be used to develop learning environments with the potential for training, technology alone does not ensure that effective learning will result.[7] In fact, few efforts have focused on how to best use these systems to establish effective environments to support learning.[8]

Technology to support learning must facilitate the development and maintenance of the competencies (i.e., behaviors, cognitions) necessary to perform required tasks. Technology must support the employment of systematic, deliberate approaches for: (a) all phases of training development (e.g., planning, preparation, execution, analysis), (b) performance measurement, and (c) feedback.

One training technology that has been suggested to develop such a learning environment as well as problem solving expertise in teams is simulations.[9] [10] Simulations are ideally suited for the training of problem solving because they allow for the exposure of teams to the types of environments and problems that they might confront in a naturalistic setting. In addition, the use of computers to develop and present simulations allows for a high level of stimulus control and structured repetition.[11] Simulations can allow the instructional developer to manipulate and pre-specify the presentation of and relationship between critical environmental features. Within a simulated environment, it also is possible for the team to conduct problem solving in a realistic manner. For example, a problem solving team could implement alternatives as well as monitor their execution.[12] However, for this training environment to be effective it not only requires a high fidelity representation of the problem solving domain, but a systematic instructional methodology as well.[13]

In an effort to gain a better understanding of how to establish an effective learning environment for Navy air wings, an investigation was performed with a goal of defining potential training needs and strategies. The following section describes the approach used for the investigation, data that were collected via critical incidents and questionnaires, and the results from the analyses based on the data.

Approach And Results

Critical Incident Data

A critical incident data base was developed by direct observations of line flights for five carrier air wing training detachments at an instrumented range. Across the detachments, over 1400 critical incidents were documented. Incidents related to loss of team or individual SA during mission execution (e.g., spillouts, timeline errors, missed tactical calls) as well as, coordination behaviors were identified. The antecedents of these problems/incidents were then traced to planning and briefing issues and observed performance trends across training. SME/Instructors provided identification of effective/ineffective performance from planning, briefing, and mission execution. SMEs also coded the critical incidents into relevant functional areas.

Critical Incident Category	Count
Delivery of Bombs on Target	29
Performance of Target Area Tactics	27
Egress from Target During Mission	18
Bomb Hit Assessment During Mission	16
Appropriate Selection of Weapon Given Target During Planning	16
Planning Contingencies for Egress	15
Planning for Bomb Hit Assessment	14
Staying on Mission Timeline During Execution	13
Planning to Ensure Acquisition of Target	13
Provide Mutual Support During Execution	6
Planning to Maximize Use of Air Wing's Capabilities	6
Awareness of Enemy Ground Threats During Execution	5

Table 1. Frequency of Critical Incidents for Striker Element

Critical Incident Category	Count
Execution of Fighter Tactics	13
Implementation of Contingencies During Mission Execution	13
Planning of Fighter Tactics	10
Briefing of Fighter Tactics	10
Adherence to Briefed Plan During Execution	8
Planning Flow Control Points	8
Staying on Mission Timeline During Execution	6
Staying in Assigned Position During Execution	6
Handling of Leakers During Mission	5

Table 2. Frequency of Critical Incidents sfor Swing Fighter Element

The database was examined to identify what individual level skills could potentially benefit from training in simulators. Tables 1-4 provide a summary of the critical incidents collected across the five carrier air wings for each of the four main elements (i.e., swing fighters, strikers, SEAD element, and command and control element). The incidents were clustered into general categories. While there is some overlap in the critical incidents categories that the elements exhibit, in general each functional area requires different competencies, and therefore, tends to demonstrate negative behaviors in its own unique types. Only those critical incident categories that occur at least five times within the critical incident database were included in the table below.

Critical Incident Category	Count
Planning for the Use of HARM	49
Effectiveness of Jamming During Execution	21
Effectiveness of HARM During Execution	17
Accounts for Enemy Ground Threats During Planning	14
Staying on Mission Timeline During Execution	8
Effectiveness of HARM During Execution	8
Utilize Good Planning Process	7
Contingencies for Filling in Aircraft Loss During Execution	7
Staying in Assigned Position During Execution	6
Appropriate Number of Call Signs for Mission	5
Planning Flow Control Points	5
Planning to Ensure Effective Jamming	5

Table 3. Frequency of Critical Incidents for SEAD Element

The results in each of these tables suggest that most of the elements difficulties are element specific. For example, within the strike element most of the critical incidents were associated with delivering bombs, target area tactics and egress from the target area. This suggests that the individual air crews could benefit from greater opportunities to practice and refine those skill that are necessary to effectively execute their own portion of a strike mission. Simulators and simulations could provide these opportunities.

While simulators and simulations could provide increased opportunities to practice many skills identified, others do not lend themselves to effective use of simulators and simulations. For example, in the swing fighter element (Table 2) and suppression of enemy air defense (SEAD) element (Table 3), incidents related to planning and briefing were observed on a relatively frequent basis. This type of training is likely to be more effectively accomplished in a classroom environment rather than a simulator. However, the presence of the incidents also suggests that when simulation is used instructors should provide time to allow trainees to practicing planning and briefing.

Interestingly, within the command and control element of the air wing the majority of critical incidents (Table 4) is in skills areas dealing with planning (e.g., battle space management, number of frequencies) rather than execution skills (e.g., maintaining the big pig picture, handling leakers). These results suggest that simulators may be less effective for training this air wing element as compared to other elements.

Critical Incident Category	Count
Planning for Battlespace Management	19
Maintaining Big Picture During Mission	17
Appropriate Number of Frequencies for Mission	8
Handling of Leakers During Mission	7
Appropriate Number of Call Signs for Mission	5

Table 4. Frequency of Critical Incidents for Command and Control Element

While the above data and discussion highlight the need for simulators and simulations for individual air crew training, these training strategies may be most effective when air wing integration is the goal of training. As noted earlier, there are limited opportunities for the air wing to practice the skills and acquire the knowledge necessary for integration. Table 5 displays the critical incidents recorded that were due to the interaction between two elements as well as the general categorization of the critical incidents. These categorizations are of three general types: Strategy Development (planning, contingencies, and briefing), Single Element Impact (fighter, battlespace management, strike, and SEAD), and Multiple Element Impact (communication and coordination).

Examination of Table 5 shows that the greatest difficulty in integration occurs between the swing fighter element (i.e., F/A-18) and the command and control element (i.e., E2-C). The interaction between these two elements is substantial across all three general types. Within Strategy Development, integration seems to be most negatively impacted by the development of contingency plans. The frequencies within this general type suggest that the two elements may possess different frameworks for thinking about the task and as a result have difficulty coordinating their activities. During execution, the integration problems are more problematic for the swing fighter element than the command and control element. This may suggest that swing fighter element is experiencing more difficulty adapting than the command and control element.

	Strategy Dev			**Single Element Impact**				**Command & Control**	
	Planning	Contingencies	Briefing	Fighter Tactics	Battlespace Management	Strike Tactics	SEAD Tactics	Communications	Coordination
Swing Fighter -- Command & Control	23	44	36	65	32	0	0	57	40
Swing Fighter -- Striker	8	2	6	12	0	5	0	0	13
Swing Fighter -- SEAD	3	0	2	0	0	0	12	1	19
Command & Control -- Striker	0	0	0	0	0	0	0	0	0
Command & Control -- SEAD	2	27	0	0	3	0	50	4	1
Striker --SEAD	0	0	0	0	0	0	0	0	0

Table 5. Frequency of Critical Incidents for Interacting Elements

The integration between the SEAD element and the command and control element appears next most difficult. Again, the development of contingency plans appears to cause the greatest difficulty in the Strategy Development section. This may differentially impact the performance of the two elements as demonstrated by SEAD element incurring the greater number of critical incidents. While both the SEAD and swing fighter element are more negatively impacted by the interaction, the impact on the command and control element could not be assessed.

In summary, the results from this data collection effort suggests that the greatest impact to improve training could be made by linking F/A-18 and E2-C simulators together. This would provide them an opportunity to practice integration skills together prior to detachment to a live flight training event. In comparison, other tasks (e.g., planning) may only be peripherally improved through the use of simulation.

Questionnaire Data

A training assessment questionnaire was developed to examine the subjective opinions of aviators on the air wing training. For theoretical reasons three areas were hypothesized to be important for understanding air wing training: mental models, team processes, and phases of training. A questionnaire with 24 items was developed to examine these three areas. The questionnaire was administered to aviators from three carrier air wings.

Preliminary examination of the items suggested that many were highly correlated with one another and that it might be possible to collapse the items into a relatively small number of constructs prior to conducting analysis. Two of the items from the training phases area were dropped from the factor analysis because of their utility as outcome measures. Both of these items measured the extent to which the execution of missions was beneficial to the participants. A principal components extraction with varimax rotation was performed on the remaining twenty two (22) items from the questionnaire. Five factors were extracted from the data set of 101 participants. The factor loadings for the individual items for these five factors are displayed in the Table 6.

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Mental Models					
Mission Types	0.869	-.025	0.007	0.047	-.030
Own Aircraft's Role	0.891	0.196	0.055	-.027	0.033
Asset Relationships	0.856	0.088	0.003	0.144	-.153
Mission Characteristics	0.756	0.277	0.025	0.228	0.210
Other Platform's Capabilities	0.659	0.281	-.031	0.201	-.078
Other Squadron's Procedures	0.485	0.426	0.207	0.239	0.029
Implementation					
Tactical Skills	0.054	0.803	0.003	0.112	0.244
Decision Making	0.232	0.704	0.196	0.174	-.033
Situation Awareness	0.400	0.675	0.228	0.089	-.038
Communication	0.201	0.658	0.092	0.407	-.090
Debrief Mission	-.014	0.508	0.190	-.285	0.324
Orientation					
Planning Missions	-.132	0.061	0.814	0.176	-.091
Briefing Missions	0.081	0.195	0.707	-.214	0.340
Integration	0.231	0.331	0.562	0.316	0.089
Adaptation					
Contingency Planning	0.157	0.091	0.292	0.702	0.217
Threat Awareness	0.276	0.369	-.104	0.677	0.119
Individual Skills					
Pre-Training Prep	0.011	0.141	0.194	0.184	0.613
Personnel Familiarity	0.308	0.149	0.302	-.001	-.533
Flight Discipline	0.409	0.208	0.174	0.237	0.439

Table 6. Factor Loadings for Exploratory Factor Analysis

The five factors identified in the table provide insight into air wing processes that were not included in the initial questionnaire construction. The first factor, labeled Mental Models, contains six of the seven items that were intended to measure that construct. The only mental model item that did not load heavily on the first factor was familiarity with air wing personnel. This result, while not expected, is explainable by the nature of the task domain. In large teams, the familiarity with individual team members may be unimportant for an individual's mental representation of the task.

The second factor was labeled Implementation, because the four team process items that loaded heaviest on this factor were skills that were most important when the team was performing a task for which the team was primarily responsible (e.g., fighters). The one training phase item that did not load heavily on this factor was the feedback that the team received after completion of that performance (i.e., mission debrief). The four team process items that did load as heavily on this factor (integration, contingency planning, threat awareness and flight discipline) differ from the four items that did (situation awareness, tactical skills, communication and decision making), in that they are less action oriented and more introspective.

The third factor, labeled Orientation, contained those items that would allow the team's different components to coordinate their actions during performance. Two of the items were training phases (planning and briefing) and the third item was a team process (integration). These items reflect the team's requirement to form a common understanding of the specific requirements for each mission they must execute.

While the third factor seeks to establish a prescription for how a performance period should occur, the forth factor, Adaptation, includes those items that determine how the team will respond when those prescriptions fail to match the task characteristics. The two items that loaded heaviest on this factor are two team processes contingency planning and threat awareness. Both of these items are measures of the extent to which the team was monitoring its environment and considering how it might change.

The final factor, Individual Skills, is a collection of items that did not load heavily on any of the other factors and appear to be primarily associated with a specific team member's interface with the larger team. While pre-training preparations and flight discipline both loaded heavily on this factor in a positive direction, personnel familiarity loaded heavily in a negative direction. This suggest that there may be some trade-off for each team member in being proficient in his/her role and interacting with other team members.

The five factors identified in the factor analysis were used in a series of MANOVAs to determine the extent to which demographic data could predict factor loadings. The first demographic variable examined was the participant's aircraft platform. Ratings were obtained from aviators from five types of aircraft (F/A-18, F-14, EA-6B, ES3 and E-2C). A Hotellings multivariate test of significance obtained an approximate F = 2.739; p < 0.01. This result suggests that participants from each of the platform types would have different loadings on the five factors. Examination of the univariate F-tests shows that two of the factors, Orientation [F(4,96) = 5.873; p < 0.01] and Adaptation [F(4,96) = 3.812; p < 0.01] were reliable. The mean values for the factors for the five platform types is displayed in Table 7.

A least significance difference (LSD) test was done to identify the differences between the platforms. The results of these tests indicated that aviators from the ES-3 platform were significantly different from EA-6B, F-14 and F/A-18 but not E-2C, with respect to their rating of the Orientation factor. These means suggest that aviators from the ES-3 platform felt that they had benefited less in the area of Orientation than the aviators from the other platform types. Results from the LSD test for the Adaptation factor indicate that the EA-6B platform was significantly different from all of the other platform types. This suggests that the training that the aviators received with respect to Adaptation was only beneficial for EA-6Bs aviators.

	F/A-18 (N=25)	F-14 (N=20)	EA-6B (N=16)	ES3 (N=23)	E-2C (N=17)
Mental Mod	0.015	0.251	0.093	0.065	-.353
Implement	0.361	0.095	-.307	0.186	-.285
Orientation	0.364	0.196	0.428	-.721	-.194
Adaptation	-.206	-.294	0.793	0.078	-.203
Indiv. Skills	-.193	0.118	-.311	0.068	0.167

Table 7. Mean Factor Scores by Platform

	1st Quart. (N=27)	2nd Quart. (N=25)	3rd Quart. (N=24)	4th Quart. (N=24)
Mental Models	0.271	0.238	0.040	-.644
Implementation	-.041	-.207	0.078	0.158
Orientation	-.095	0.006	-.216	0.305
Adaptation	-.355	0.147	0.139	0.052
Individual Skills	-.209	0.276	0.094	-.116

Table 8. Mean Factor Scores by Experience

The second MANOVA examined the extent to which the number of hours in a platform affected factor loadings. This was the first of two additional analyses that examined the extent to which experience affected factor loadings. The aviator participants were divided into quartiles based on their experience in their current platform. Those in the first quartile had between 45 and 300 hours in their current platform. Those in the second quartile had between 301 and 700 hours, the third 701 and 1100 hours, and finally, aviators in the fourth quartile had between 1101 and 2600 hours experience. A Hotellings multivariate test of significance obtained an approximate F = 1.889; $p < 0.05$. This result suggests that participants with differing levels of experience would have different loadings on the five factors. The univariate F-tests shows that only Mental Models changed with respect to experience level differences [F(3,96) = 5.046; $p < 0.01$]. The results show that those with the fewest hours in platform rated the benefit received in the area of Mental Models as the highest. This suggests that the usefulness of training for improving mental models decreases with experience. Table 8 provides the mean factor loadings for the four quartiles.

The third MANOVA conducted used the number of deployments that an aviator had been on as a measure of experience. Participants were divided into three groups: those who had never been deployed, those who had been deployed on a carrier once, and those who had been deployed 2 or more times. A Hotellings multivariate test of significance obtained an approximate F = 2.817; $p < 0.01$. These results, like the previous analysis, show that participants with differing levels of experience have different loadings on the five factors. The univariate F-tests shows that not only was the Mental Models factor significant [F(2,96) = 5.408; $p < 0.01$], but also the Adaptation factor [F(2,96) = 4.182; $p < 0.01$]. As with the previous analysis, this analysis suggests that the usefulness of training for improving mental models decreases with experience. However, as aviators gain more experience, their ratings suggest that they are benefiting more, and are better able to adapt to the dynamic environment in which they operate. Table 9 provides the means for the these analyses.

Two additional analyses were conducted after the completion of the MANOVAs. The five factors were regressed onto the two items that measured the extent to which the execution of missions was beneficial. The first of these two items examined the extent to which the training received during execution was beneficial to the participant as an aviator. The second item measured the extent to which the training received during execution was beneficial in increasing the air wing's integration. The overall $\underline{R}^2$ was significant for both the aviators own proficiency [$\underline{R}^2$ (5,92) = 0.339; $p < 0.01$] and for air wing integration [$\underline{R}^2$ (5,92) = 0.364; $p < 0.01$]. Examination of the Betas in Table 10 show that for both of the regression equations, the Implementation, Orientation and Adaptation factors independently predict a significant portion of the variance in ratings of execution benefits. In addition, in the air wing integration regression, the Mental Model factor is also significant. This suggests that while mental models did not have a significant effect on an aviator's individual performance, they are critical for large team performance.

The results of the data collected highlight the importance of attending to cognitive factors when training. The importance of knowledge to integrative team performance suggests that fidelity of scenarios should be a driving factor when designing simulators and simulations. In addition, because of their importance, simulators should be instrumented to capture information that will aid in the assessment of mental models and other cognitive structures. In addition to emphasizing the importance of cognitive factors, these data also demonstrate the criticality of adaptability for effective team performance. These issues should be considered when developing scenarios for simulators.

	0 Deploy (N=24)	1 Deploy (N=41)	> 2 Deploy (N=34)
Mental Models	0.456	0.038	-.360
Implementation	-.216	-.123	0.264
Orientation	-.107	-.075	0.157
Adaptation	-.492	0.222	0.035
Individual Skills	-.066	0.205	-.152

Table 9. Mean Factor Scores by Number of Aviator Deployments

	Mental Models	Implementation	Orientation	Adaptation	Individual Skills
Own Proficiency	0.105	0.466**	0.213*	0.233**	0.090
Integration	0.174*	0.388**	0.388**	0.180*	-.026

* $p < 0.05$; ** $p < 0.01$

Table 10. Beta Weights for Regression of Factor Loadings on to Performance Scores

Summary

The goal of the current effort was to conduct a preliminary investigation directed towards identifying potential training needs and strategies for large tactical team training. Specifically, the effort sought to understand the knowledge, skills, and attitudes that need to be learned and trained and how to employ simulation technology in support of this learning. A conceptually derived framework for describing the coordination requirements for Navy 'air wings was developed. While the framework was preliminary in nature, it provided a mechanism for developing the data collection tools and organizing the results.

The results provided important insight as to how different elements within the large tactical team differed in terms of what competencies were required. If competencies can be identified, then it may be possible to make better decisions with regard to supplementing large tactical team training with simulation. The methods employed in the current effort provided a mechanism to begin to differentiate where and when to use simulation resources.

Although the results provide insight for the potential utility of simulation and simulator for air wing training, clearly additional research is required. For example, research should be conducted to assess the extent to which the use of simulation by individuals prior to air wing training results in more effective performance.

It should also be noted that efforts to understand the training requirements for large tactical teams is a complex task. The size and dynamic nature of the environment presents considerable challenges (e.g., needs analysis, measurement, data aggregation). While the focus of the current effort was on Navy air wing teams, the methods, strategies, and tools have applicability to analyze the potential use of simulation for other types of large distributed tactical teams (e.g., Joint Task Forces, Emergency Response Teams). As training resources become more valuable, the need to maximize the effective use of live and simulated training opportunities for large tactical teams will continue to increase in importance. Methods, such as those demonstrated in the current effort will facilitate decisions for the effective selection and use of training resources and technologies (e.g., simulation).

References

1 Orasanu & Connolly (1993). The reinvention of decision making. In G. A. Klein, J. Orasanu, R. Calderwood & C.E Zsambok (Eds.), Decision making in actions: Models and methods (pp. 3-20). Norwood, NJ: LEA

2 Oser, R. L., McCallum, G. A., Salas, E., & Morgan, B. B., Jr. (1989). Toward a definition of teamwork: An analysis of critical team behaviors (Tech Rep. 89-004). Naval Training Systems Center, Orlando, FL.

3 Cannon-Bowers, J. A., Salas, E., & Converse, S. A. (1993). Shared mental models in expert team decision making. In N. J. Castellan, Jr. (Ed.), Individual and group decision making: Current issues (pp. 241-246). Hillsdale, NJ: LEA.

4 Kleinman, D. L. & Serfaty, D. (1989). Team performance assessment in distributed decision making. Proceedings of the Symposium on Interactive Networked Simulation for Training (pp. 22-27). Orlando, FL.

5 Driskell, J. E., Olsen, D., Hays, R. T., & Mullen, B. (1995). Training decision-intensive tasks: A constructivist approach (Tech. Rep. 95-007). Orlando, FL: Naval Air Warfare Center, Training Systems Division.

6 Bell, H. H. (1995). The engineering of a training network. Proceedings of the seventh annual International Training Equipment Conference, Arlington, VA, 365-370.

7 Salas, E., & Cannon-Bowers, J. A. (1997). Methods, tools, and strategies for team training. In M. A. Quiñones & A. Ehrenstein (Eds.), Training for a rapidly changing workplace: Applications of psychological research (pp. 249-279). Washington, DC: APA Press.

8 Dwyer, D. J., Fowlkes, J. E., Oser, R. L., Salas, E., & Lane, N. E. (1997). Team performance measurement in distributed environments: The TARGETs methodology. In M. T. Brannick, E. Salas, & C. Prince (Eds.), Team performance assessment and measurement: Theory, methods, and applications (pp. 137-153). Hillsdale, NJ: LEA.

9 Cannon-Bowers, J. A., & Bell, H. R. (1997). Training decision makers for complex environments: Implications of the naturalistic decision making perspective. In C. Zsambok & G. Klein (Eds.), Naturalistic decision making (pp. 99-110). Hillsdale, NJ: LEA.

10 Means, B, Salas, E., Crandall, B. & Jacobs, T. O. (1993). Training decision makers for the real world. In G. Klein, J. Orasanu, R. Calderwood, & C. E. Zsambok (Eds.), Decision making in action: Models and methods (pp. 306-326). Norwood, NJ: Ablex.

11 Oser, R. L., Cannon-Bowers, J. A., Dwyer, D. J., & Salas, E. (1997). Establishing a learning environment for JSIMS: Challenges and considerations [CD-ROM]. Proceedings of the 19th annual Interservice/Industry Training, Simulation and Education Conference, Orlando, FL, 144-153.

12 Gualtieri, J., Parker & Zaccaro (1995). The effect of individual and group characteristics on group

decision making. Paper presented at the Society for Insustrial/Organizational Psychology Annual Conference, Orlando, FL.

13 Oser, R. L., Cannon-Bowers, J. A., Dwyer, D. J., & Salas, E. (1997). An event-based approach for training: Enhancing the utility of joint-service simulations. Paper presented at the 65th Military Operations Research Society Symposium, Quantico, VA: Office of Naval Research.

AIAA-98-4540

MOBILE ON-BOARD FLIGHT SIMULATOR FOR PILOT'S PREFLIGHT PREPARATION AND TACTICAL THINKING TRAINING

Oleg A. Yakimenko*, PhD, Alexander V. Firsov, Vladimir V. Demidov, Andrey A. Efteev

Air Force Engineering Academy n.a. Prof. N.E.Zhukovskiy
40, Leningradskiy prospect, 125190, Moscow, Russia

In present paper it's considered the general ideology, hardware architecture and some aspects of mathematical foundation of special software for mobile simulator, which provides pilot's preflight training and flight mission image formation on the board of his "native" airplane with "native" habitual deskpanel and controls. Proposed idea was already realized and demonstrated during International Aviaspace saloon in Zhukovskiy, Russia in 1997 on the board of deck Sukhoy-33 aircraft (see below).

Introduction

It's well known the set of difficulties, which accompanies the process of huge volume quick-changed information reception, its storage, processing and adequate to real tactical situation decision making by pilots of modern high-maneuverable aircraft. This is the reason that flight accidents majority today occurs due to pilot's faults (without failures of material part), and furthermore - due to piloting (control actions) errors.

Interviewing of more than 200 pilots shows that most of all pilots would like to have their skills support training at operative-tactical level and level of decisions realization (including control actions and armament implementation)[1]. It's desirable for all pilots, irrespective of class, flight hours and airplane type. And of course they desire to have their piloting skills training on simulators, which are very closed to real inside- and outside cabin environment.

As the creation of specialized seminatural simulators is very difficult and expensive the present paper deals with the problem of hardware architectural solution and appropriate software creation for specialized mobile adaptive simulator (SMAS), which can assist the pilot's training before concrete flight, can help in choosing of appropriate optimal decisions, and can provide pilot's confidence in success of flight mission realization (in principle such software for flight simulator can be considered as onboard pilot intelligent support system forerunner). That is the speech goes about close to analog of embedded training[2].

SMAS configuration general description

On start position (in hangar) real airplane with working onboard equipment (electric and hydro systems but not engine) is connected with usual computer (Fig.1). Over head-up display easily removable flat graphic display is installed[#] (Fig.2). It is connected by serial plugs with onboard common trunk. The data about forthcoming flight are entered in computer. Pilot can see drawn on the screen of display outcabin three-dimensional space, targets, basic panel desk indicators, and also all graphic images of the unified indication system (UIS), appropriated to the mode of flight, to condition of arms system and dynamics of its application. So pilot can execute preflight preparation (with help of hierarchical menu system he can choose particular fragments of flight, change conditions and parameters, analyze performed training results, etc.).

SMAS applicability

SMAS is the specialized simulator of combat airplane application against the air and surface-based targets ensuring the training in full accordance with the operational instructions on flight preparation.

In difference from complex simulator crewmember behavior is fulfilled in conditions less approximate to

[*] Professor of AFEA, Corresponding Member of Aviation and Aeronautics Academy of Science, yakimenko@mailcity.com.

[#] As visualization system it's possible to use the virtual reality helmet and PC-projectors.

actual, as there is no dynamic bench and the simplified visualization system is used, however, one from the main qualities of such simulation is imitation of UIS images at various stages of aircraft and its arms system application at control with actual controls and switches.

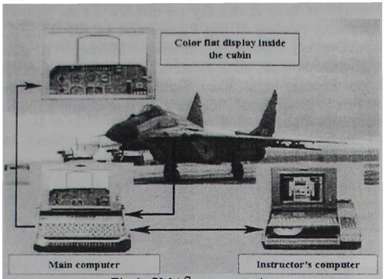


Fig.1. SMAS components

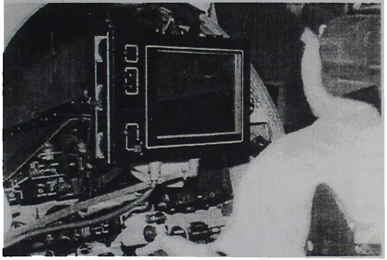


Fig.2. Inside cabin environment illustration

The training can be implemented:

- during preliminary preparation for common skills training;
- during preflight preparation for studying of forthcoming particular mission;
- at airdromes which are non-equipped with appropriate complex simulators, or at faults of simulators in case of their presence;
- on the airplanes with restricted or exhausted resource of the plane, its engines, landing gear, etc.;
- with instructor - second pilot in cabin (in case of twofold airplane) and near the plane (in monadic version);
- with group actions simulation at association of two SMAS on a number of close placed airplanes for improvement of inter-airplanes coordination;

Main quality of SMAS – it's its mobility, which should supply operative (no more than 20 minutes) installation (dismounting) of simulator elements on combat airplane and training of forthcoming mission fulfillment, allowing the pilots to maintenance very unstable and fastly lost skills of airplanes and arms application.

Not less important SMAS property is it adaptation, since it uses unified information channels of objective control system[*].

Outside cabin environment images and graphics can be adapted for actual tactical tasks of particular air subdivision and its geographical location.

Another quality of SMAS also is the capability of real time analysis of executed training as directly on airplane by pilot itself, and in any suitable room with use of the same SMAS elements (computer and visualization system) by group of pilots.

Some features of SMAS intellectualized software

SMAS software includes[3]: the algorithms and programs describing complete mathematical model of airplane movement (the object-oriented approach allows to change a type of airplane rather quickly) on a runway and in air as rigid body; the models of its subsystem (locator, head-up display, armament system, jet power system, etc.); the model of standard atmosphere and different wind disturbances; the algorithms and programs of outside cabin conditions formation and visualization, the programs of cabin desk drawing; special algorithms of pilot's intelligent support system (PISS) with the programs of optimal recommended trajectories of different maneuvers as "road-in-the-sky" on the screen of display construction[4].

This software allows to simulate dynamics of airplane spatial movement in real time scale with the high degree of affinity to real objects both simulated dynamics of the airplane, and virtual reality inside and outside cabin conditions. Thus it provides both pilotage training and goal utilization of aircraft simulation.

Mentioned PISS algorithms provide short-cut time scale different optimizational tasks solution with help of several «fast» modifications of direct method of variational calculus[5]. This ideology is implemented, for example, for such more or less long-term maneuvers as: take-off and climbing, flight on a route, surface-based target attack, guidance on moveable object[6], aircombat, descent and landing.

It's supposed that all particular tasks (except en-route flight[7]) have been preliminary formalized in order to form the grid of calculation nodes; then in this nodes variational trajectory task by means of appropriate modification of direct method at large stationary computers have been solved; obtained during solution defining referred near-optimal trajectories parameters

[*] To the present time SMAS was established on three modifications of Fulcrum airplane) (monadic, twofold and deck), and also on the transport aeroplane, and arisen problems of program interface were solved for 20-30 minutes.

(only two or three for each trajectory!) have been storaged in the image of "zero approximation to near-optimal trajectories bank".

During real flight preparation for real strategic and/or tactic environment (from which boundary conditions and phase coordinates restrictions are defined) for real aircraft configuration (with accounting of its subsystems and even pilot current state - SMAS software allows to simulate the set of different faults and damages) that gives us the set of controls limitations, concrete bank is being chosen, following multi-dimensional interpolation is being executed and with help of determinated from "trajectories bank" zero approximation concrete optimizational task is being solved in accelerated scale of time (it takes approximately 5% from trajectory time itself for Pentium-133 processor). Reconstructed by this way trajectory is presented to pilot in cognitive image of road in the sky on display for its following tracking (analog of Big Picture and PCAD formats of McDonnell Douglas, Boeing Military Airplane, Thompson-CSF, etc.). Road is equipped with appropriate wayside signs like "Thrust-up", " Thrust-down", "Fire", "Drag flaps", "Gears", etc. See examples below.

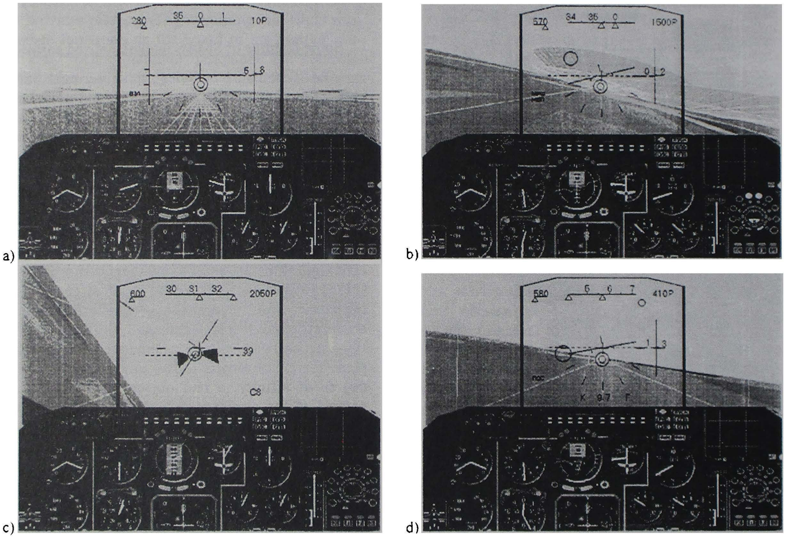


Fig.3. Examples of piloting at different stages of flight with road-in-the-sky assistance (take-off (a), enroute flight (b), surface-based target attack approach (c), descent and landing (d))

The utilization of road-in-the-sky track idea for pilot's training permits to have a qualitatively new conditions for control skills obtaining - pilot see all trajectory up to the final point[8].

For example Fig.3 shows the set of trajectories obtained during particular maneuvers surface-based target attack fulfillment skills training with gradual decrease of tracking errors and sufficient increase of efficiency of following armament application. Fig.5 demonstrates the example of concrete trajectory of abrupt prelanding descent with the turn in condition of strong wind disturbances presence tracking.

Economical estimation approach

Lets perform particular analysis concerning only one aspect of armament application skills maintenance.

During 8-10 minutes at preflight preparation with help of SMAS pilot can execute any exercise on arms application up to 5 times, and to obtain quantitative estimation of misses received at simulation of shooting from the gun, launching of controlled and unguided rockets and bombing.

Considering, that entire flight mission of combat airplane terms approximately 45 minutes, and that as a rule in one flight it's performed no more than two approaches on target attack (that is actually on control of arms system it's expended no more than two minutes), it's possible to make the conclusion, that 8-10 minutes activities of pilot with SMAS at the stage of preflight preparation is equivalent to 4-5 flights or 3-3.75 hours of airplane flight operation.

If as estimated cost of one hour of flight operation of modern high-maneuverable airplane is taken 20 thousand dollars, the training of one pilot with SMAS use in mentioned condition can be appreciated as economies in 60-75 thousand dollars (for particular types of airplanes this estimation can vary from 40 up to 80 thousand dollars).

At the stage of preliminary preparation, which is executed during all working day, the pilot with help of SMAS can perform different exercises on goal aircraft application practically without any time constrains, training up his skills on particular exercise up to automatisms, and putting itself in more and more difficult conditions.

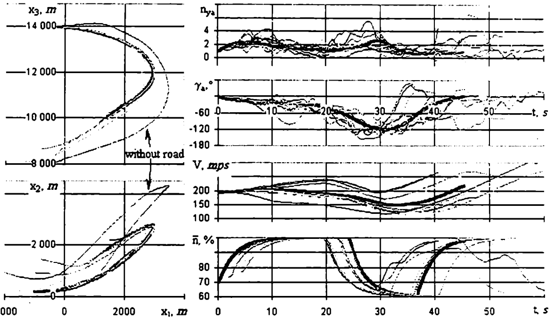


Fig.4. Examples of afterflight analysis

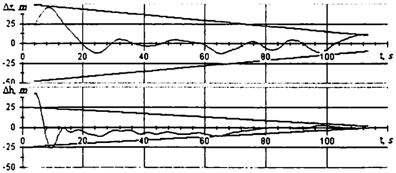


Fig.5. Illustration of reference road-in-the-sky tracking during abrupt descent with the turn (figure shows aligned road)

Considering 16-20 minute activity of each pilot with SMAS as equivalent to 8-10 flights, for eight hour working days 12 pilots of one squadron in view of several 20 minutes breaks, can save 1.3 - 1.6 million dollars. And pilots squadrons of regiment - 3.9-4.8 million dollars accordingly.

Thus submitted above calculations take into account only consumption of aircraft resources (plane, engines, fuel, etc.) though economic profit from obtained experience especially received in conditions, which cannot be created really without risk for crew-members lives (application of radio electronic counteraction means, low-altitude flight, etc.) can appear to be much greater.

The own costs of SMAS operation (functioning of 2-3 computers and visualization system) with costs on airplane ground operation are incommensurable. At training on the one, specially allocated for this purpose airplane, for stipulated early training conditions of the pilots of one squadron, 3.2-4.0 hours of airplane's systems resources will be expended. In case of SMAS utilization on the planes personally fixed for each pilot the resources consumption of each airplane, will decrease to original 16-20 minutes accordingly.

Conclusions

Performed up to present time investigations of developed mobile on-board adaptive flight simulator for pilot's preflight preparation, tactical thinking training and combat implementation skills obtaining and maintenance on different multi-regime aircraft justify the possibility of appropriate software design, and efficiency of its utilization. It's supposed also that SMAS in the whole can be rather competitive if to take into account efficiency/cost ratio criteria.

References

1. *Yakimenko O.A.* Pilot requirements to universal airborne intelligent pilot decisions support system // Proceedings of NAECON. - Dayton, OH, 1995.
2. *Huges R.G.* Aircrew embedded training: a logical extension of current and emerging technologies // Proceedings of NAECON. – Dayton, OH, 1986.
3. *Yakimenko O.A., Demidov V.V., at al.* High visualization capacities flight minisimulator with demonstrated prototype of operator motor action support system // Proceedings of AIAA FSTC. - San Diego, CA, 1996.
4. *Yakimenko O.A.* "Road in the sky" as the means of pilot's motor actions during maneuvering intelligent support // Proceedings of NAECON. - Dayton, OH, 1996.
5. *Yakimenko O.A.* Shortcut-time spatial trajectories onboard optimization and their cognitive head-up display vizualisation for pilot's control actions during maneuvering support // Proceedings of ICIASF. - Monterey, CA, 1997.
6. *Yakimenko O.A., Dobrokhodov V.N.* Airplane trajectory control at the stage of rendezvous with maneuvering object algoruthmes sinthesis // Proceedings of NAECON. – Dayton, OH, 1998
7. *Yakimenko O.A., Alekhin D.V.* Modification of direct method for flight route and profile at presence of prohibited zones optimization // Proceedings of AIAA AFM. - Boston, MA, 1998.
8. *Theunissen E.* Integrated design of a Man-Machine Interface for 4-D Navigation. PhD Thesises. - Delft University Press, 1997.

AIAA-98-4541

USING MODELING AND SIMULATION IN THE TARGET VALIDATION AND ACCREDITATION PROCESS

Brian Gravelle
Navy Target/Threat Validation Project Director
Naval Air Warfare Center, Weapons Division
Point Mugu, CA 93042

Patrick Burris
Lead Systems Engineer
Sverdrup Naval Systems Group
760 Paseo Camarillo, Suite 300
Camarillo, CA 93010-0799

Abstract

The Department of Defense (DoD) has instituted a requirement for formal validation and accreditation of target vehicles used for critical weapon system testing and training. The validation process, known as target-to-threat validation, constitutes comparing the selected target to a specific threat to determine how well it represents the threat characteristics that are critical for the intended use of the target. The results of this comparison are then used during accreditation to determine if the target is suitable for the specific test purposes.

Since threat vehicles can rarely be measured directly, modeling and simulation (M&S) is being used extensively in performing the validation and accreditation processes. In particular, M&S is used for predicting characteristics of the target and threat to evaluate the effect that these target-to-threat differences will have on the target's intended use. These models can take the form of simple data tables, detailed mathematical functions, or complex simulations that define target, threat and weapon system behavior. [1]

Background

Requirement for Target Validation & Accreditation

Major weapon system acquisition decisions are based largely on the results of live fire testing against target vehicles. For those decisions to be informed and correct, the targets must be sufficiently representative of threats to ensure that the testing has revealed the real world capabilities of the weapon. This process of determining threat "representativeness" is addressed through target validation and accreditation (V&A).

Validation is defined in the Department of Defense (DoD) regulations as "the process of determining the manner and degree to which a target is an accurate representation of a threat from the perspective of the intended uses of the target." Validation does not determine whether or not a target is acceptable for a specific purpose, rather it is simply a comparison of those characteristics of the target and the threat that may impact the tests for which that vehicle may be used. Accreditation is defined as "the official certification that a target is acceptable for use for a specific purpose." It is during accreditation that the differences between a target and a threat are examined in detail to determine how those differences will impact the test or training exercise. This impact is then weighed against the test objectives to determine if the target can serve as an adequate substitute tfor the threat.

Historically, the weapon system program offices were responsible for V&A of their targets. This process was typically performed on an informal basis and was inconsistent among the various weapon systems programs. Shortfalls within the test community such as a target's inability to properly represent a threat or a lack of target/threat data were not always properly communicated back to the responsible organizations. A review by the DoD Inspector General's (IG) office in 1991, published in IG Report 92-020, indicated:

[1] This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

"DoD had not validated the extent aerial targets replicated the performance capabilities or signatures of threat missiles and aircraft during operational testing and training, and it had not quantified target limitations...recommend DoD regulations be revised to include a validation process for aerial targets and the material internal control weaknesses be reported and tracked."

As a result of this report, DoD regulations were changed to require validated targets to pass certain program milestones. A three-year Centralized Test and Evaluation Investment Program (CTEIP) project was initiated in 1995 to address the issues outlined in the IG report. The goal of this project was to establish and proof a common validation process for the all of the services and the Ballistic Missile Defense Organization (BMDO). This CTEIP program is currently in its final phase of execution.

The Target V&A Process

The CTEIP project has resulted in the establishment of an approved validation process. This process consists of the following basic steps:

1. Based on the intended use of the target, such as crew training or missile system testing, a set of parameters called the Target Critical Characteristics (TCC) are derived.

2. Values for the TCC for both the target and the threat are researched and compared to determine the target-to-threat differences.

3. A brief assessment of the impact of target-to-threat differences on the target's intended use is prepared.

4. Target-to-threat differences and the impact assessment are combined to form the validation report.

Once validation is completed, the accreditation process examines the specific tests planned for the target and uses the data gathered during validation to conduct a detailed impact assessment. This assessment evaluates how differences between the target and the threat will impact individual test results and judges whether these impacts will affect successful completion of the test objectives. Based on this assessment a decision is made regarding the target's suitability to represent the threat.

Application of M&S to Target Validation

Requirement for M&S

Often target and/or threat TCC cannot be measured directly due to project cost and/or asset availability. Modeling and simulation (M&S) must be used to bridge the gap between the measurable parameters and the complete data set necessary to characterize the target/threat. Models used in target validation typically consist of data sets that describe attributes of the target/threat, for example, the coefficients that describe the target's aerodynamic performance are referred to as the aerodynamic model. Simulations are used with these models to compute time varying performance data, such as maneuver response times or maximum attainable speeds. A typical simulation used for target V&A is a digital trajectory program. This simulation uses models of a target's aerodynamic, guidance, autopilot, and control surface properties to estimate the target's kinematic performance.

Derivation of TCC

The first step in performing target V&A is identification of the TCC. The TCC list is derived based on the threat to be represented and the intended use of the target. A sample list of TCC for a target that is intended to represent an aerial threat vehicle for purposes of live-fire missile testing is provided in table 1.

TCC Computation

Once identified, values for the TCC for both the target and the threat must be determined. These values may be determined via direct measurement or by using M&S tools. The applicability of M&S varies dependent on the type of parameter being researched.

Physical Size/Vulnerability

Physical size parameters are typically measured directly if the vehicle of interest is available; otherwise they can be derived from photographs and scale drawings. Vulnerability is more difficult to quantify, particularly since the vulnerability of a particular component is based not only on its own composition and physical location but also on the size and speed, and approach angle of the fragment that is striking it. For validation purposes, vulnerability is normally limited to identification of critical components and their physical location.

Table 1. Target Critical Characteristics

Category/Characteristic
Physical Size/Vulnerability
Fuselage length
Fuselage diameter
Wing span
Location of critical components
Vulnerability of critical components
Kinematic Performance
Maximum flyout range
Operational altitude profile
Operational speed profile
Maneuver type(s)
Maneuver response time
RF Signature
Radar cross section
Jet engine or propellor modulation effects
Polarization characteristics
Spatial location and amplitude of dominant RF scatterers
IR Signature
Radiant Intensity
Radiance

Kinematic Performance

If available, kinematic performance is derived from the design specification documents and operational flight test data. If these sources are either unavailable or inadequate, flight performance simulations are used for predicting kinematic performance. A block diagram of a typical flight performance simulation is displayed in figure 1. These simulations incorporate detailed models of the target/threat's aerodynamic properties, autopilot functions, and flight control systems. These simulations require validation and accreditation procedures independent of the target V&A process.

RF Signature

In the case of sub-scale targets it is usually cost effective to directly measure the characteristics of the RF signature using static tests (if the vehicle is available). For large-scale targets, however, static measurement costs increase dramatically due to the effort required to modify and manipulate the vehicles. Dynamic measurements, i.e. in-flight measurements, are cheaper to obtain but more difficult to precisely measure and calibrate. In these cases it may be best to directly measure a small set of data and supplement it with predicted data using M&S tools.

RF signature data can be predicted using a variety of computer simulations. These simulations model the physics involved in bouncing radar signals off surfaces and require a detailed description of the physical surface of the target vehicle. This surface description is provided in the form of a facet model, a mathematical description of the target composed of many simple geometric shapes such as triangles or polygons.

IR Signature

Direct measurement of a target's IR signature is typically more difficult than characterization of its RF

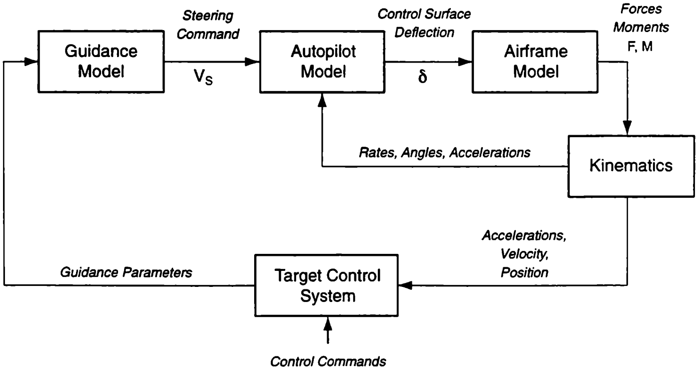


Figure 1. Flow Diagram of a Target Kinematic Model

signature. While RF signature effects can be largely determined in a static environment, IR measurements must be taken under dynamic conditions. Additionally, IR properties vary widely due to atmospheric and background effects. Dynamic measurements of IR signature are usually restricted to a few test points under non-ideal conditions. To obtain a broad characterization of a target's signature under a variety of atmospheric and background conditions, flight test data is used to validate a computer model. This computer model consists of a facet model of the target combined with an IR signature prediction code such as the Spectral In-Band Radiometric Imaging of Targets and Scenes (SPIRITS). Once the computer model is validated, various backgrounds and atmospheric conditions can be simulated without the need to perform additional flight testing.

Application of M&S to Target Accreditation

Requirement for M&S

Accreditation of a target requires insight into the impact that target-to-threat differences have on test results. Typically, this requires insight into how the unit-under-test or personnel perform during the test. Since the threat is rarely available for insertion into a test exercise and cost considerations severely limit the number of tests that can be conducted, M&S is often heavily relied on to simulate the test conditions. If the test environment can be simulated, models of the target and the threat can be alternately inserted into the simulation to analyze their impact on the test results. Once these impacts are identified they can be evaluated to determine a) the target-to-threat difference that resulted in the impact and b) their net effect on successful execution of the test objectives.

Such an approach to accreditation was recently performed in support of a weapon system flight test program and is illustrated in figure 2. As shown in the figure, detailed models of the target and threat were developed from data produced during the validation process. These models were then injected into detailed simulations to assess the impacts of target-to-threat differences on missile performance. More detailed descriptions of some of these models and simulations follows.

Target/Threat Models

RF Signature

The M&S tools that are used during validation to predict RF signature are typically not suitable for use during accreditation due to runtime speed requirements, software implementation, or data limitations. RF prediction simulations such as XPATCH require far too much processing time to dynamically compute the RF return of a target for use in a missile simulation. As a result additional models must be developed. Since these models will now be used to represent the target or threat they must be validated against the original data produced during validation.

RF signature models used in missile simulations often consist of a series of statistical functions which model the reflected RF amplitude as a rayleigh function with an aspect dependent mean and variance. Angular scintillation, or glint, is sometimes modeled as a gaussian function. With the advent of highly sophisticated RF receivers, however, these techniques are often inadequate to properly portray the characteristics of a complex RF signature. More advanced models, such as the N-point model, are being used more widely. The N-point model consists of a set of RF scatterers that are positioned spatially around the target such that their vector sum exhibits behavior similar to that observed during the target measurement process. N-point models can be implemented to run at rapid speeds and can compute amplitude and angular scintillation effects more accurately than can simple statistical models. Unlike statistical models, N-point models can also be used to dynamically recalculate the RF return for systems whose range gate does not always encompass all of the scatterers located on the target.

Kinematics

Kinematic models used during accreditation are typically similar to those used during validation, but are of a lower fidelity to support the runtime requirements of the performance simulation. An alternative approach to using the low fidelity models is to generate a series of trajectory profiles using the detailed target simulation and, using a time dependent lookup function in the missile performance simulation, interpolate acceleration, speed, and position data from the target profile as needed.

Vulnerability

Vulnerability models are similar to the facet models used for generating RF and IR signatures, however, vulnerability models include far more internal details of the target such as the fuel tanks, hydraulic lines, and major structural components. These models are used in missile lethality simulations to determine which, if any, of the target/threat's vulnerable components are destroyed when the missile's warhead detonates.

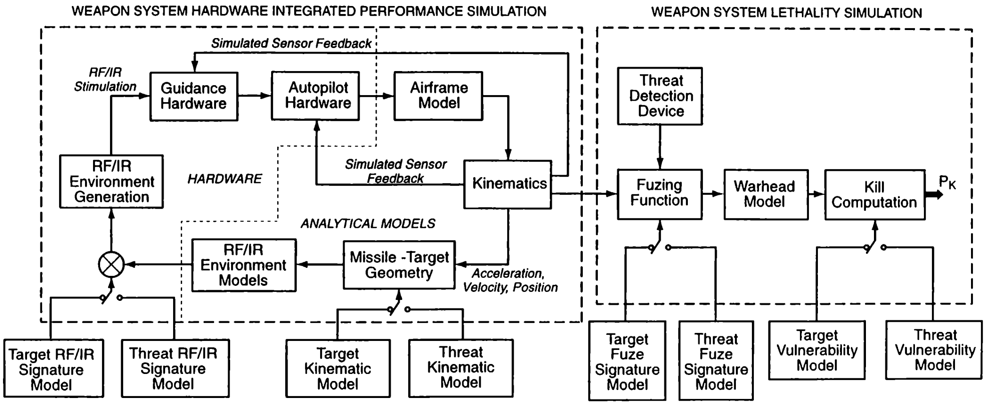

WEAPON SYSTEM HARDWARE INTEGRATED PERFORMANCE SIMULATION
WEAPON SYSTEM LETHALITY SIMULATION
Simulated Sensor Feedback
RF/IR Stimulation
Guidance Hardware
Autopilot Hardware
Airframe Model
RF/IR Environment Generation
HARDWARE
Simulated Sensor Feedback
Kinematics
ANALYTICAL MODELS
RF/IR Environment Models
Missile -Target Geometry
Acceleration, Velocity, Position
Target RF/IR Signature Model
Threat RF/IR Signature Model
Target Kinematic Model
Threat Kinematic Model
Threat Detection Device
Fuzing Function
Warhead Model
Kill Computation
P_K
Target Fuze Signature Model
Threat Fuze Signature Model
Target Vulnerability Model
Threat Vulnerability Model

Figure 2. M&S Implementation of Accreditation Analysis

The lethality simulation contains detailed models of the missile's warhead to deterministically calculate the speed and trajectory of each warhead fragment. Impacts of warhead fragments with vulnerable components of the target are tallied in a probabilistic fashion to determine the likelihood of target kill. Figure 3 illustrates a target vulnerability model being used in a lethality simulation.

Summary

The value of target V&A is to ensure that weapon system test results will reflect the ability of the weapon system to perform in real world scenarios against real world threats. To support V&A, detailed descriptions of the target and threat must be provided. Direct measurement of target and threat parameters can be costly or difficult for a variety of reasons. M&S is often relied upon to fully characterize a vehicle in the absence of directly measured data.

An assortment of M&S tools is commonly used. During validation, these tools consist of models and simulations that predict the characteristics of the target/threat vehicles. During accreditation this scope is increased to include weapon system performance simulations to enable the user to ascertain the impacts of target-to-threat differences on planned testing. The quality of target V&A is thus closely linked to the quality of the M&S tools employed.

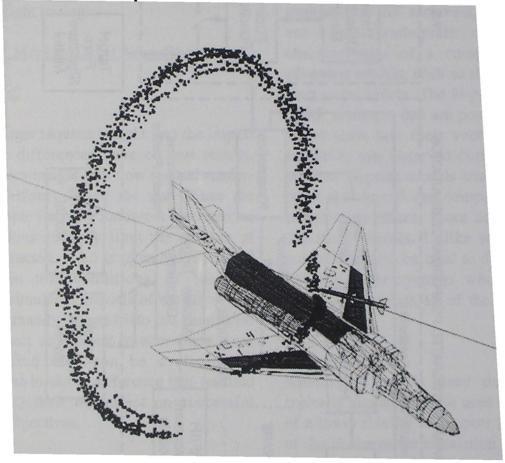


Figure 3. Illustration of Vulnerability Model in a Lethality Simulation

AIAA-98-4542

IN-LINE, BANK-TO-TURN AUTOPILOT FOR SIMULATIONS USING INTERNET SOFTWARE AND POLE PLACEMENT

William J. West II
Associate Principal Engineer
Sverdrup Technology
214 Government Avenue
Niceville, Florida 32578

Abstract

An in-line, bank-to-turn (BTT) autopilot constructed using the pole placement technique has been implemented in a six degree-of-freedom simulation. By in-line, it is meant that the autopilot calculates its gains as the simulation progresses with time. The BTT autopilot requires aerodynamic derivatives, angles, angular rates, and moments of inertia to generate a linearized approximation to the nonlinear equations of motion. The controller uses the linear approximation, the pole placement algorithm, and desired eigenvalues to calculate the gains at the desired sample frequency. The software in the pole placement subroutine was constructed from free software downloaded from the Internet. The full-state feedback design, with proportional plus integral compensators, provides the desired transient and steady-state responses. The fully coupled linear system is eighth-order with three inputs and five outputs. Discussion of the BTT logic and underlying mathematics is presented in this paper. Plots from simulated flights steered by two guidance laws are included. These test simulations of a direct attack munition demonstrate the performance and versatility of this ideal approach and its usefulness in studying preliminary concepts.

Nomenclature

α =angle of attack or Alpha defined as atan(w/u)
$\dot{\alpha}$ =derivative of angle of attack
β =sideslip angle or Beta defined as atan(v/u)
$\dot{\beta}$ =derivative of sideslip angle
cos =cosine

$C_{l\beta}$ = change in roll moment due to Beta

$C_{l\delta_p}$ = change in roll moment due to aileron

$C_{l\delta_r}$ = change in roll moment due to rudder

C_{lp} = change in roll moment due to roll rate

C_{lr} = change in roll moment due to yaw rate

$C_{m\alpha}$ = change in pitch moment due to Alpha

$C_{m\dot{\alpha}}$ = change in pitch moment due to change in Alpha

$C_{m\delta_q}$ = change in pitch moment due to elevator

C_{mq} = change in pitch moment due to pitch rate

$C_{n\beta}$ = change in yaw moment due to Beta

$C_{n\delta_p}$ = change in yaw moment due to aileron

$C_{n\delta_r}$ = change in yaw moment due to rudder

C_{np} = change in yaw moment due to roll rate

C_{nr} = change in yaw moment due to yaw rate

$C_{N\alpha}$ = change in normal force due to Alpha

$C_{N\dot{\alpha}}$ = change in normal force due to change in Alpha

$C_{N\delta_q}$ = change in normal force due to change in elevator

C_{N_q} = change in normal force due to pitch rate

C_{Y_β} = change in side force due to Beta

$C_{Y_{\delta p}}$ = change in side force due to aileron

$C_{Y_{\delta r}}$ = change in side force due to rudder

C_{Y_p} = change in side force due to roll rate

C_{Y_r} = change in side force due to yaw rate

d = reference length

δ_p = aileron

δ_q = elevator

δ_r = rudder

g = gravity

I_{XX} = axial moment of inertia about X axis

I_{XY} = product of inertia

I_{XZ} = product of inertia

I_{YY} = axial moment of inertia about Y axis

I_{YZ} = product of inertia

I_{ZZ} = axial moment of inertia about Z axis

p = roll rate

$\dot{p}$ = derivative of roll rate

ϕ = Euler roll angle

Q = dynamic pressure

q = pitch rate

$\dot{q}$ = derivative of pitch rate

r = yaw rate

$\dot{r}$ = derivative of yaw rate

S = reference area

sin = sine

θ = Euler pitch angle

u = component of velocity along X axis

v = component of velocity along Y axis

V = velocity magnitude

w = component of velocity along Z axis

W = weight

Introduction

One of the time-consuming and more difficult challenges facing every conceptual munition is the design of the control system. Not only is the task laborious, but for every change in the aerodynamics or mass properties of the vehicle, the entire control system design process may need to be repeated. It appears desirable to have an automated way to generate the control system for simulations of these concepts that utilize the perfect information contained in the input files. By having this automated autopilot, it is possible to perform a multitude of studies that include maximum range, footprints, navigation improvements, and guidance law sensitivities without regenerating predefined gain schedules.

The best way to study munition enhancements and make performance predictions is to acquire the actual control system (if it is available) and implement it into the simulation. This, however, may constitute a bottleneck situation since the necessary control system data may not be available, the design may be incomplete, or company-proprietary software restrictions may apply. This situation tends to make thorough research into new capabilities for these weapon systems difficult. From a modeling and simulation perspective, this new approach will provide representative, albeit perfect, time-domain behavior by which the control system deficiencies have been removed and the new technology can be studied in the context of a more mature munition system.

This paper begins with a top-level view of the methodology followed by a flow chart of the control algorithm and input parameters. The web site address from where LAPACK pole placement subroutines can be downloaded is included. Plots of command and achieved states versus time for two guidance laws are also provided. The mathematical details of the implementation and a simplified example are reserved for the end of the paper.

Top-level View of Implementation

The interface diagram for guidance and control loop of the simulation having the in-line BTT autopilot is shown in Figure 1.

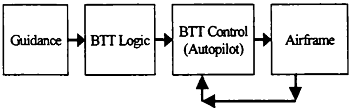


Figure 1. Interface diagram

The basic blocks include the guidance, BTT logic, BTT autopilot, and airframe model. The guidance block outputs lateral and normal acceleration commands. These guidance commands are converted by the BTT logic into commands for angle-of-attack, sideslip angle, roll rate, pitch rate, and yaw rate. The sideslip command is always zero. These five commands from the BTT logic are the reference signals that feed the BTT autopilot. Angle-of-attack, sideslip, roll, pitch, and yaw rates are the states from the airframe model and are shown feeding back to the BTT autopilot. These states are used to create an error signal from the reference. They are also used to build the upper five-by-five partition of the augmented system matrix. The autopilot converts the difference between the reference signals and the states into aileron, elevator, and rudder commands that feed a fin-mixing scheme. The output of the fin-mixing scheme drives the actuator model. The actuator model then simulates fin deflections creating changes to or maintaining the motion of the airframe. The process repeats until termination.

The BTT logic[1] as shown in Figure 2 converts the acceleration commands into the angle-of-attack and angular rate commands. The acceleration commands from guidance are adjusted to account for the effects of gravity. If the guidance law performs this gravity adjustment before the BTT logic is enabled, then this step would be bypassed. The gravity-compensated commands are then fed into the vertical and roll channels. In the vertical channel, the normal acceleration command is scaled to create a desired normal force coefficient. This coefficient and the Mach number serve as the independent variables to a table look-up of angle-of-attack commands that will trim the vehicle model for the independent variables. The angle-of-attack command is then limited.

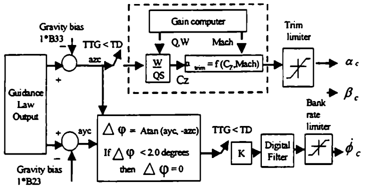


Figure 2. BTT Logic block diagram

The roll channel takes the commanded acceleration components and forms a change in the bank angle. A dead band of two degrees prevents the munition from continually reacting to low frequency noise or minor adjustments. The gain ‘K’ converts the change in the bank angle into a bank angle rate that is filtered and limited to give the final bank angle rate. The digital filter for this exercise was a single pole low-pass filter. TTG (time-to-go) and TD (time disable) are time variables that shut down the BTT logic. This shutdown allows the angle-of-attack and rates to zero before impact. TTG may be estimated by any of several means, and TD is set in mission planning according to the time constant of the munition. Table 1 gives the parameters of the roll channel that were used for this paper.

Altitude > 25,000 feet	K=0.25 second^{-1}
5,000 feet < Altitude ≤ 25,000 feet	K=0.50 second^{-1}
Altitude ≤ 5,000 feet	K = 1.0 second^{-1}
Cutoff filter frequency	6.2 radians /second
Bank rate limiter	6.5 radians /second
Time-to-go	0.4 seconds

Table 1. BTT Logic roll channel parameters

Like the roll channel, the vertical channel table and limits are aerodynamic specific. The angle-of-attack limit is approximately 24 degrees while the C_Z table covers a range of values for both positive and negative angles-of-attack.

Two more features of the BTT logic do not appear in Figure 2. If the bank rate command were of opposite polarity from the current roll rate, then the gain ‘K’ could be increased by a factor of two. By feeding the roll rate ‘p’ into the BTT logic and comparing its sign to that of the bank rate command prior to the digital filter, a decision to scale by two can be made. This scaling helped to minimize overshoot and delays in following sign changes to the roll angle change command. The second feature of the BTT logic that is not shown in Figure 2 is the transformation of $\dot{\phi}_c$ into body coordinates. Although the commanded accelerations are in body coordinates, the bank angle change is in velocity coordinates. To transform the commanded accelerations to velocity coordinates prior to the calculation of the bank angle change would require the multiplication of the cosine of angle-of-attack by the normal acceleration command to create normal acceleration in velocity coordinates. This cosine would decrease the normal acceleration by a maximum of nine percent. The transformation to body coordinates from velocity coordinates and the resulting roll, pitch, and yaw commands are

$$p_c = \dot{\phi}_c * \cos\beta * \cos\alpha \tag{1}$$

$$q_c = \dot{\phi}_c * \sin\beta \tag{2}$$

$$r_c = \dot{\phi}_c * \cos\beta * \sin\alpha \tag{3}$$

As shown in Figure 3, the angle-of-attack, sideslip, roll, pitch, and yaw rate commands are fed into the autopilot as the reference signals. Five differences are formed between the commands and the states. Because of the sign reversal at the final summing junction caused by the negative in the control law, the sign of the difference between the command and state is opposite from that expected. Also, the three proportional plus integral (PI) compensators that are necessary to obtain zero steady-state error for a constant input are shown. Only three PI compensators can be used because of system controllability. Because the output of the autopilot will be aileron, elevator, and rudder commands, the number of compensators is limited to the channels that these three commands influence. The chosen design proved to be superior to other locations for the PI compensators (i.e., p, q and r channels or p, alpha, and r channels). The compensated and uncompensated differences are then multiplied by gains determined by the pole placement algorithm. There is a set of eight gains for each of the three fin deflection commands. The gains are numbered to comply with the states from (4) that the gains serve as multipliers.

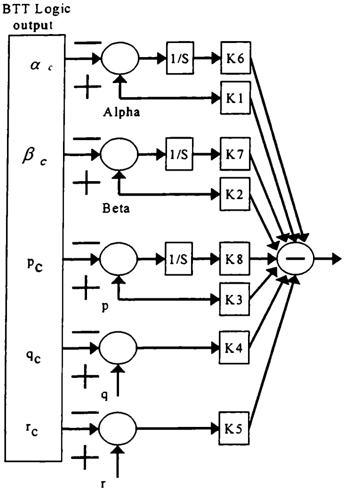


Figure 3. Autopilot block diagram

The discussion of the contents of the $A_{5X5}{}^2$ matrix and the $B_{5X3}{}^2$ are deferred to the appendix. These augmented matrices are discretized and provided to the pole placement algorithm.

$$\dot{X} = \left[\begin{array}{c:c} A_{5X5} & 0_{3x3} \\ \hdashline I_{3x3} & 0_{5x3} \end{array}\right] \begin{bmatrix} \text{Alpha} \\ \text{Beta} \\ p \\ q \\ r \\ \int \text{Alpha} - \alpha_c \\ \int \text{Beta} - \beta_c \\ \int p \quad - p_c \end{bmatrix} + \left[\begin{array}{c} B_{5X3} \\ \hdashline 0_{3x3} \end{array}\right] u - \begin{bmatrix} 0_{5X5} \\ \alpha_c \\ \beta_c \\ p_c \end{bmatrix} \tag{4}$$

The state vector is modified slightly before calculation of the control. This modification, shown in Figure 4, consists of subtracting q_c from q, r_c from r, and performing Euler's integration for states six, seven, and eight. States one through five are created from the nonlinear equations of motion.

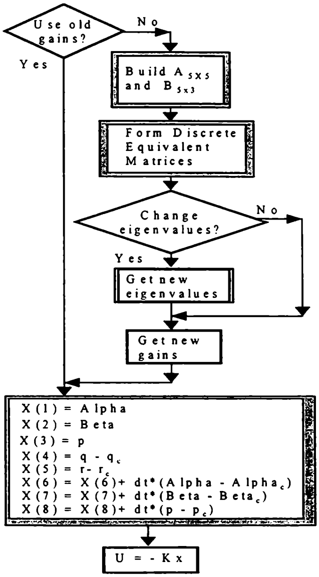

Use old gains?
No
Yes
Build A_{5x5} and B_{5x3}
Form Discrete Equivalent Matrices
Change eigenvalues?
No
Yes
Get new eigenvalues
Get new gains
X(1) = Alpha
X(2) = Beta
X(3) = p
X(4) = q - q_c
X(5) = r- r_c
X(6) = X(6)+ dt*(Alpha - Alpha_c)
X(7) = X(7)+ dt*(Beta - Beta_c)
X(8) = X(8)+ dt*(p - p_c)
U = - Kx

Figure 4. Flow chart of BTT autopilot

Figure 4 contains the flow chart of the autopilot process. A decision to use the old or new gains is made on entry. This decision can be a timer such as every 0.01 seconds, or a counter that counts the passes through the subroutine. If new gains are desired, the A_{5X5} and the B_{5x3} are created and loaded into the augmented system and input matrices. A discrete time equivalent of these augmented matrices is then formed. A decision block to change the eigenvalues is then encountered. Changing the eigenvalues can occur as a function of altitude (which was the case for the plots in the Results section), dynamic pressure, or some other flight condition where speed or damping requirements change. The selection of the eigenvalues can be a two-week exercise in itself due to the nonlinearities that exist in the control law. That is, there is nothing in the control law requiring smooth transition in the commands. Between each time step, the angle-of-attack may change several degrees. Slow eigenvalues prevent oscillations in the control; however, they suffer from responsiveness when the commands rapidly change. Table 2 contains the s-domain eigenvalues that were converted to discrete time equivalent values for the test cases.

Altitude above 25,000 feet	Altitude between 5,000 and 25,000 feet	Altitude below 5,000 feet
-3.5+/-3.0j	-9.5+/-4.5j	-8.5+/-3.5j
-4.0+/-3.5j	-9.5+/-4.5j	-8.5+/-4.0j
-4.25+/-4.0j	-9.75+/-5.0j	-8.75+/-4.5j
-5.0	-10.0	-9.5
-5.5	-10.5	-10.0

Table 2. Eigenvalues for test cases

The discrete-time, augmented system and input matrices along with the eigenvalues are input into the pole placement subroutine. Notice the linear system is not propagated in this approach, but the modified state vector from (4) does contain Euler's integration of the PI compensators and the differences that were discussed. This state vector for the full-state feedback system is multiplied by the gains giving the negative of aileron, elevator, and rudder commands.

Notice in Figure 4 that the BTT logic is not shown. There are two logical places where the BTT logic could fit. One would be after the guidance has determined the normal and lateral acceleration commands. Since these commands are fixed for perhaps ten times longer than the autopilot frequency, it would make some sense to place the BTT logic **outside** the autopilot loop. Another possible location for the BTT logic would be **inside** the autopilot loop. Since there is no compensator on the roll angle change command; and since the roll, pitch, and yaw rates commands are a function of the current angle-of-attack and sideslip, it might make more sense to include the BTT logic as part of the autopilot process flow. The selection is dependent on the needed accuracy and speed of response. For the results presented, the BTT logic was included in the autopilot flow whenever the gains were updated.

The explanation of the theory behind the algorithm for computing the gains is found in Reference 3. With this paper and MATLAB's place subroutine[4] m-file as a model, the pole placement gain calculation subroutine was constructed. Singular value decomposition, QR decomposition, and solutions to linear equations subroutines were found using LAPACK software. A good web site for LAPACK subroutines can be found at

http://www.netlib.org/lapack/index.html

with lapack/complex16 and lapack/double directories having the bulk of free software that must be assembled.

Overview of IMAGINE

The Integrated Methodology for the Assessment of GPS/INS Enhancements (IMAGINE)[5] has been in development for four years by the Air Force Research Laboratory at Eglin AFB, Florida. IMAGINE is the simulation used to implement the in-line, BTT autopilot. It has been used to support a variety of activities relating to the Global Positioning System (GPS) and antijam research. As shown in Figure 5, IMAGINE includes models for GPS, jammers, antenna, rotating earth, target, and munition. The GPS constellation can be created from a downloaded ephemeris file from a NAWC Chinalake web site located at

http://sirius.chinalake.navy.mil.

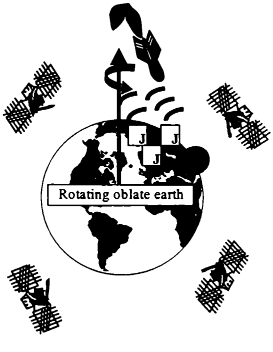


Figure 5. IMAGINE diagram

The ephemeris files are updated on the hour. This feature of the simulation allows for realistic satellite positions. A default satellite constellation is also available. Jammer models include antenna patterns, gain, frequency, and pointing. The rotating earth model allows for simulated flights anywhere in the world and provides information for the navigation and navigation error algorithms. These algorithms along with a vast array of error models allow for simulation of integrated inertial navigation systems. The subsystem interactions are accounted for in IMAGINE in that the navigation Kalman filter corrections are used to update the guidance before commands are calculated for the autopilot. Winds can be loaded through the input file. IMAGINE has an F-16 silhouette model used for track-under-wing predictions of GPS. IMAGINE was also used to study jamming scenarios for semi-dynamic ground tests of an antijam antenna affixed to the top of a bus driven through a jamming field. Future plans include further development of the antijam model.

Results

To test the methodology, an aerodynamic database was generated from a schematic diagram of an experimental flight test vehicle (FTV) of a 2000-lb, MK-84 bomb. Expected mass properties for the symmetrical airframe were also available. Telemetry from a flight test was located and used to select the parameters that were given earlier.

Figure 6 is the plot of the ground trace of flight test trajectory, the simulated trajectory using an optimal guidance law, and the simulated trajectory using proportional guidance. The FTV is known to use BTT control driven by an optimal guidance law that tries to minimize the control deflections for the entire flight; therefore, it is most desirable for IMAGINE with the in-line BTT autopilot to closely match that trajectory. Careful inspection of figure 6 shows that the two traces differ by several hundred feet for the first part of the trajectory, but are in agreement in position during later times. Part of the difference is attributed to IMAGINE generating its acceleration commands with truth data versus imperfect navigation data. The option of using erroneous navigation and inertial measurement data and Monte Carlo processing is one of IMAGINE's capabilities not required for this exercise. The proportional guidance trajectory is included to show that the autopilot was not tuned to just one trajectory. As seen in Figure 6, the proportional guidance trajectory is very different from that simulated and flown on the mission.

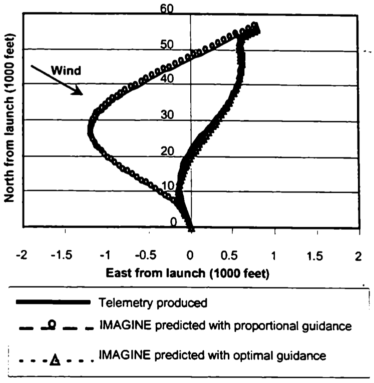


Figure 6. Ground trace comparisons

Figure 7 contains the plots of simulated angle-of-attack and roll rate. Both command and actual values are plotted against time. The vertical scale is correct for both degree and degrees per second time series. Angle-of-attack trims at approximately 20 degrees and zeros prior to impact. The roll rate 'p' undergoes a 24-degree-per-second change after the initial three-second delay required for safe separation.

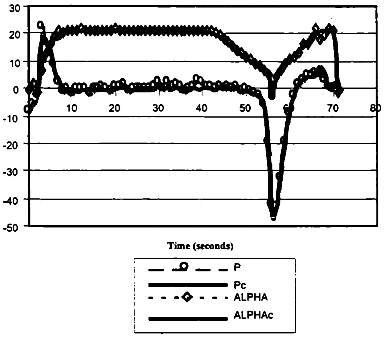


Figure 7. Simulated angle-of-attack and roll rate using optimal guidance.

The achieved states follow the commanded for both angle-of-attack and roll rate. Telemetry plots are not included in Figure 7 since they have the same form and peak values that occur approximately five seconds sooner in flight. This time difference can be attributed to using true values or slightly better aerodynamics in the model than seen with the actual FTV. Efforts to change drag did not significantly affect the five-second difference.

By plotting angle-of-attack and roll rate on the same plot, the reader can see the effects of optimal guidance and minimization of control. Because the guidance law tries to minimize the sum of the square of the commanded control deflections, the elevator is dominant during the first part of the flight while the roll channel remains essentially dormant. Prior to initializing corrective roll, the elevator is commanded to minimum value while the autopilot places an emphasis on roll control. Notice that once the roll requirements are satisfied, the elevator command is again given preference. Finally, both channels are driven to zero less than 0.5 seconds before impact.

The same simulated release condition and target location was attempted with proportional guidance instead of optimal guidance law. The ground trace for this simulated flight was shown in Figure 6. A plot of the angle-of-attack and roll rate history is shown in Figure 8. These time series illustrate the autopilot's ability to follow a different set of commands without additional adjustments.

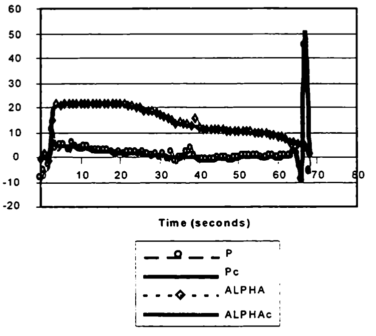


Figure 8. Simulated angle-of-attack and roll rate using proportional guidance

The BTT autopilot's ability to track lateral acceleration commands is a difficult task to evaluate. Maintaining satisfactory control of the bank angle change command is not a requirement of the BTT autopilot nor is tracking the actual lateral acceleration command. There will always be some steady-state error in yaw rate since the controllability issue prevents the incorporation of an additional compensator. Therefore, besides the roll rate performance previously shown, the consistency with which the BTT autopilot maintains a zero degree sideslip angle is the other metric used to gauge lateral control performance. Figure 9 shows the results of sideslip angle versus time. This plot demonstrates the autopilot's capacity to maintain the near zero requirement regardless of the guidance law. The initial non-zero values occur prior to control activation. It should also be noted that the test case includes 100 nautical mile per hour winds from the northwest that were modeled in the simulation.

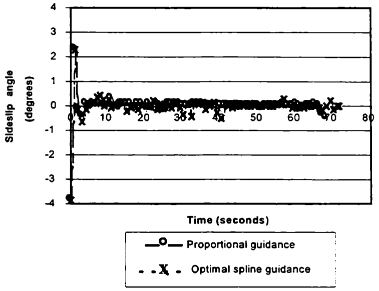


Figure 9. Sideslip time series

Normal acceleration performance is easier to measure. Recall from the BTT logic discussion that normal acceleration is transformed into trim angle-of-attack commands. If this table were exact and the guidance did not command more capability than the munition could achieve, then the plot of commanded and achieved normal acceleration would agree. Figure 10 presents the achieved and commanded normal accelerations for the optimal guidance simulated flyout. At first, it appears that there may be an error in the transformation of normal acceleration command into angle-of-attack command in the BTT logic since the achieved and the commanded differ prior to 40 seconds. However, inspection of Figure 7 shows that the munition is achieving its maximum angle-of-attack prior to this time. Also, this maximum angle agrees with that from the telemetry for the flight test. In reality, the difference is created because the optimal guidance is trying to maintain maximum altitude exceeding the munition's capability. Negative acceleration is up in body coordinates. Also note that once the guidance law relaxes its maximum capability requirements, the accelerations are in agreement except for the one perturbation during final maneuver.

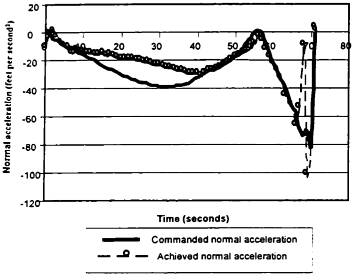


Figure 10. Normal accelerations for the simulated optimal guidance flight

Figure 11 contains the normal acceleration time series for the proportional guidance simulation. Notice the smaller requirements on normal force and thus, the time series are in agreement. Referring back to figure 8, near maximum angle-of-attack requirements are demanded; however, these requirements map into approximately two-thirds of a 'g' of acceleration. The incongruity beginning at the 50-second time mark is attributed to a mismatch between the acceleration command and the resulting angle-of-attack command output from the table in the BTT logic. As seen in Figure 8, the angle-of-attack and its command are in agreement during this time. Nevertheless, the acceleration difference is on the order of five feet per second^2.

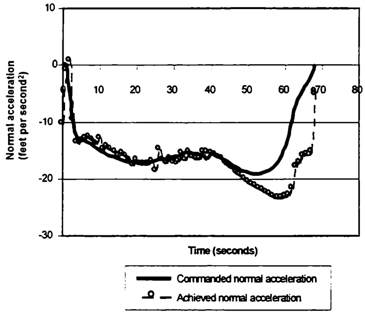


Figure 11. Normal acceleration for proportional guidance

Run time requirements are surprisingly small for this approach. IMAGINE is currently hosted on a 266 Mega Hertz (Hz) Pentium II processor personal computer with an ASUS motherboard. To integrate its derivatives, IMAGINE uses a fourth-order Runge-Kutta scheme. There are many levels of complexity from which the user can select, however, the level used for these test cases and timing study is the most basic. That is, GPS, jamming, antenna model, error model, and Kalman filter were all disabled. Only the 30 states that contribute to the continuous dynamics were integrated. The required states include the six for the equations of motion, two for each actuator, and sixteen for the true navigator. The autopilot was executed at 100 Hz with relinearization at 50 Hz. Guidance was executed at 10 Hz. Integration occurred at a 400 Hz rate. The simulation performed each simulated flight in approximately real-time.

Although the purpose of the paper is to present the capabilities of the in-line, BTT autopilot, the report would be incomplete without a table of the terminal impact conditions as a function of the guidance law. Table 3 presents the terminal miss distance, flight path angle, angle-of-attack, and time of flight for the two guidance options.

	Optimal guidance	Proportional guidance
Miss distance (feet)	14.0	2.0
Flight path angle (degrees)	-80	-49
Final angle of attack (degrees)	-0.4	1.6
Time of flight (seconds)	71	68

Table 3. Terminal condition

The optimal guidance tries to maximize final flight path angle and minimize final angle-of-attack. Caution should be used regarding conclusions relative to miss distance since the optimal guidance meets the circular error requirements and attempts to reduce angle-of-attack at the expense of increased miss distance. Additionally, the terminal conditions for the optimal guidance are in agreement with those from the flight test that was used to do the quasi-validation.

Conclusion

The in-line, BTT autopilot has been shown to provide representative BTT control and utility. The aerodynamic database generated by Missile DATCOM was not compared with data from wind tunnel measurements. With the digitally-created aerodynamic database and the in-line, BTT autopilot, representative flyouts can be made and used to study enhancements such as new filters, guidance laws, or antijam systems. Flight test predictions, however, are another matter since the control system is not the one that is actually flown. The utility, as far as predictions for flight tests are concerned, is that if the actual control system behaves ideally, its flight and state behavior should resemble that predicted with the in-line autopilot.

Acknowledgements

The author would like to acknowledge Mr. Randy Jones, AFRL/MNGG branch manager, Vy Nguyen AFRL/MNGG FAEA, James Moore, and Eddie Gibbs also of AFRL at Eglin AFB, Florida for their sponsorship and patience during the research and development of the in-line, BTT autopilot.

Appendix

The equations used to fill the A_{5x5} and B_{5x3} arrays are fully developed in Reference 2. This section contains a summary from that report. The basic idea is to linearize non-linear equations at each operating point for which a new set of gains is desired. The operating point vector consists of the current angle-of-attack, sideslip, roll, pitch, and yaw rates, plus the aileron, elevator, and rudder deflections. The linearization procedure[6,7]consists of taking partial derivatives of each of five nonlinear equations with respect to each quantity in the operating point vector. These partial derivatives are evaluated at the operating point and stored in the augmented system and input matrices to create the linear system. Five equations are used instead of six since control along the longitudinal axis is ignored. The five nonlinear equations are described in (5) through (9).

$$\dot{\alpha} = \left(1 - \frac{gQS\, C_{N_{\dot{\alpha}}}}{WV} \right)^{-1} \left[(q - p\beta) + \frac{gQS}{WV}\left(C_{N_\alpha}\alpha + \frac{d}{2V} C_{N_q} q + C_{N_{\delta_q}} \delta_q \right) + \frac{g}{V}\cos\theta\cos\phi \right] \tag{5}$$

$$\dot{\beta} = p\alpha - r + \frac{gQS}{WV}\left(C_{Y_\beta}\beta + \frac{d}{2V} C_{Y_p} p + \frac{d}{2V} C_{Y_r} r + C_{Y_{\delta_p}} \delta_p + C_{Y_{\delta_r}} \delta_r \right) + \frac{g}{V}\cos\theta\sin\phi \tag{6}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} I_{XX} & -I_{XY} & -I_{XZ} \\ -I_{XY} & I_{YY} & -I_{YZ} \\ -I_{XZ} & -I_{YZ} & I_{ZZ} \end{bmatrix}^{-1} *$$

$$\begin{bmatrix} QSd\left(C_{l_\beta}\beta + \frac{d}{2V} C_{l_p} p + \frac{d}{2V} C_{l_r} r + C_{l_{\delta_p}} \delta p + C_{l_{\delta_r}} \delta r \right) + (I_{YY} - I_{ZZ})qr - I_{YZ}\left(r^2 - q^2 \right) + I_{XZ}pq - I_{XY}rp \\ QSd\left(C_{m_\alpha}\alpha + \frac{d}{2V} C_{m_{\dot{\alpha}}}\dot{\alpha} + \frac{d}{2V} C_{m_q} q + C_{m_{\delta_q}} \delta q \right) + (I_{ZZ} - I_{XX})rp - I_{XZ}\left(p^2 - r^2 \right) + I_{XY}qr - I_{XZ}qr \\ QSd\left(C_{n_\beta}\beta + \frac{d}{2V} C_{n_p} p + \frac{d}{2V} C_{n_r} r + C_{n_{\delta_p}} \delta p + C_{n_{\delta_r}} \delta r \right) + (I_{XX} - I_{YY})pq - I_{XY}\left(q^2 - p^2 \right) + I_{YZ}rp - I_{XZ}qr \end{bmatrix}$$

(7)
(8)
(9)

Partial derivatives of these five non-linear equations are then formed and evaluated at the operating point to form the desired matrices.

$$A_{5x5} = \begin{bmatrix} \frac{\partial\dot{\alpha}}{\partial\alpha} & \frac{\partial\dot{\alpha}}{\partial\beta} & \frac{\partial\dot{\alpha}}{\partial p} & \frac{\partial\dot{\alpha}}{\partial q} & \frac{\partial\dot{\alpha}}{\partial r} \\ \frac{\partial\dot{\beta}}{\partial\alpha} & \frac{\partial\dot{\beta}}{\partial\beta} & \frac{\partial\dot{\beta}}{\partial p} & \frac{\partial\dot{\beta}}{\partial q} & \frac{\partial\dot{\beta}}{\partial r} \\ \frac{\partial\dot{p}}{\partial\alpha} & \frac{\partial\dot{p}}{\partial\beta} & \frac{\partial\dot{p}}{\partial p} & \frac{\partial\dot{p}}{\partial q} & \frac{\partial\dot{p}}{\partial r} \\ \frac{\partial\dot{q}}{\partial\alpha} & \frac{\partial\dot{q}}{\partial\beta} & \frac{\partial\dot{q}}{\partial p} & \frac{\partial\dot{q}}{\partial q} & \frac{\partial\dot{q}}{\partial r} \\ \frac{\partial\dot{r}}{\partial\alpha} & \frac{\partial\dot{r}}{\partial\beta} & \frac{\partial\dot{r}}{\partial p} & \frac{\partial\dot{r}}{\partial q} & \frac{\partial\dot{r}}{\partial r} \end{bmatrix} \qquad (10)$$

$$B_{5x3} = \begin{bmatrix} \frac{\partial\dot{\alpha}}{\partial\delta_p} & \frac{\partial\dot{\alpha}}{\partial\delta_q} & \frac{\partial\dot{\alpha}}{\partial\delta_r} \\ \frac{\partial\dot{\beta}}{\partial\delta_p} & \frac{\partial\dot{\beta}}{\partial\delta_q} & \frac{\partial\dot{\beta}}{\partial\delta_r} \\ \frac{\partial\dot{p}}{\partial\delta_p} & \frac{\partial\dot{p}}{\partial\delta_q} & \frac{\partial\dot{p}}{\partial\delta_r} \\ \frac{\partial\dot{q}}{\partial\delta_p} & \frac{\partial\dot{q}}{\partial\delta_q} & \frac{\partial\dot{q}}{\partial\delta_r} \\ \frac{\partial\dot{r}}{\partial\delta_p} & \frac{\partial\dot{r}}{\partial\delta_q} & \frac{\partial\dot{r}}{\partial\delta_r} \end{bmatrix} \qquad (11)$$

The $\dot{q}$ equation (8) has a $C_{m_{\dot{\alpha}}}$ term and $\dot{\alpha}$ is not a state. To form the fourth row of (10) and (11), the partial derivatives from (5) are multiplied by $C_{m_{\dot{\alpha}}}$ and other factors. The resulting products are added to the partials shown in row four of (10) and (11).

Example

Three objectives of the autopilot are to stabilize the airframe, provide desirable time- domain response, and provide zero steady-state error for a constant input. Stability is often indicated by all of the eigenvalues in the left half of the complex s-plane, or inside the unit circle in the z-plane. The time-domain response is dependent on the location of these eigenvalues. To satisfy these first two objectives, a linear combination of states, outputs, and commands can be created and applied to the airframe model as input. Zero steady-state error is accomplished by increasing the system type by adding free integrators. This objective is accomplished with the addition of compensators whose gains can be determined with those for the feedback paths.

To illustrate the concept, let the unstable transfer function be

$$G(s) = \frac{4}{s-2} \qquad (12)$$

The corresponding state and output equations are

$$\dot{x} = 2x + 4u \qquad (13)$$

$$y = x \qquad (14)$$

Because the root of the characteristic expression, which is the denominator of the transfer function, is positive, the system is unstable. It is desirable to stabilize the system and move this eigenvalue to –2. The control 'u' is formed by

$$u = -Kx \qquad (15)$$

The unknown gain 'K' is found to be the number one so that the eigenvalue for the closed loop system is placed in its desired location.

$$\dot{x} = 2x + 4(-Kx) = (2-4K)x = -2x \qquad (16)$$

However, for a constant input, the system has a steady-state gain of positive 2 as seen by the final value theorem.

$$G(s=0) = \frac{4}{0+2} = 2 \qquad (17)$$

By adding a proportional plus integral (PI) compensator, the order of the system increases resulting in a zero steady-state error for a step input. In block diagram form, a reference signal 'r' is applied to the PI compensator whose output is subtracted from the feedback of the system state multiplied by the gain 'k_1' to form the input 'u'. Figure 12 shows the diagram of the compensated system. With this structure and the required eigenvalue locations of –2 and –8, the transfer function from the reference input to the output

$$G(s) = \frac{16}{s^2 + 10s + 16} \qquad (18)$$

Again, by the final value theorem, this system has a steady-state error of zero for a constant input.

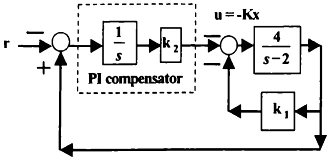


Figure 12. Compensated pole placement controller

To complete the exercise, the state equations[6] for the compensated system are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \left[\begin{array}{c|c} 2 & 0 \\ \hline 1 & 0 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \end{bmatrix} u - \begin{bmatrix} 0 \\ 1 \end{bmatrix} r \qquad (19)$$

The original plant matrix '2' appears in the upper left partition of the augmented system. The state fed back to the integrator contributes a '1' in the new system matrix. The original input matrix appears in the upper partition of the new input matrix. The third term, on the right hand side, is the negative of the reference signal. Since the autopilot will be executed at discrete intervals, discrete time equivalent matrix can be found for both the system and input matrices[7]. The discrete time form of the third term is found by multiplying the integration time step by the third term. This product integrates the negative of the reference signal using Euler's integration.

References

[1]Hirsch, I.A., Langehough, M.A., Bossi, J.A., Lee, K.L., Fix, T.C., "Advanced Robust Autopilot," AFATL-TR-89-64, Boeing Aerospace, Kent, WA, July 1989.

[2]Smith, R.L. Captain, "An Autopilot Design Methodology for Bank-to-Turn Missiles," AFATL-TR-89-49, Air Force Armament Laboratory, Eglin AFB, FL, August 1989.

[3]Kautsky, J., Nichols, N.K., Van Dooren, P., "Robust pole placement assignment in linear state feedback," *International Journal of Control*, Vol.41, No.5, May 1985, pp.1129-1155.

[4]MATLAB, The MathWorks,Inc., Natick, MA.

[5]West,W.J. II, "Simulating Anti-Jam Performance Trade-Offs on Guided Munitions Performance," Proceedings of the National Technical Meeting, Navigation and Positioning in the Information Age, The Institute of Navigation, Santa Monica, CA, January 1997.

[6]Phillips, C.L., and Harbor, R.D., *Feedback Control Systems*, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1991, pp 434-435, 598-602.

[7]Brogan, W.L., *Modern Control Theory*, Third Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1991, pp 350, 565-570.

AIAA-98-4543

MODELING AND SIMULATION OF A DIFFERENTIAL ROLL PROJECTILE*

by:

Mark F. Costello[†]
Department of Mechanical Engineering
Oregon State University
Corvallis, Oregon 97331

Abstract

This paper develops the equations of motion for a differential roll projectile configuration with 7 degrees of freedom. The dynamic equations are generated in a generic manner such that the forward and aft components are mass unbalanced. A hydrodynamic bearing exists between the forward and aft components, which couples the roll degrees of freedom. Through a simulation investigation, it is shown that bearing resistance and forward/aft body mass ratio are the dominant factors in determining the roll dynamics. For spin rates typical of fin stabilized projectiles; the trajectory is essentially independent of both bearing resistance and mass ratio.

Symbols

x, y, z : Position vector components of the center of mass expressed in the inertial reference frame.

θ, ψ : Euler pitch, and yaw angles.

ϕ_F : Euler roll angle of the forward body.

ϕ_A : Euler roll angle of the aft body.

u, v, w : Translation velocity components of the center of mass resolved in the fixed plane reference frame.

p_F : Roll axis component of the angular velocity vector of the forward body expressed in the fixed plane reference frame.

p_A : Roll axis component of the angular velocity vector of the aft body expressed in the fixed plane reference frame.

q, r : Components of the angular velocity vector of both the forward and aft bodies expressed in the fixed plane reference frame.

X, Y, Z : Total external force components on the projectile expressed in the fixed plane reference frame.

L_F, M_F, N_F : External moments on the forward body expressed in the fixed plane reference frame.

L_A, M_A, N_A : External moments on the aft body expressed in the fixed plane reference frame.

m_F : Forward body mass.

m_A : Aft body mass.

m : Total projectile mass.

$[I_F]$: Mass moment of inertia matrix of the forward body with respect to the forward body reference frame.

$[I_A]$: Mass moment of inertia matrix of the aft body with respect to the aft body reference frame.

$[I]$: Effective inertia matrix.

D : Projectile characteristic length.

C_i : Projectile aerodynamic coefficients.

q_a : Dynamic pressure at the projectile mass center.

α : Longitudinal aerodynamic angle of attack.

β : Lateral aerodynamic angle of attack.

$[T_F]$: Transformation matrix from the fixed plane reference frame to the forward body reference frame.

$[T_A]$: Transformation matrix from the fixed plane reference frame to the aft body reference frame.

c_V : Viscous damping coefficient

V : Magnitude of mass center velocity.

* Proceedings of the 1998 AIAA Modeling and Simulation Technologies Conference, Boston, MA.

[†] Assistant Professor, Member AIAA.

Introduction

Compared to conventional munitions, the design of smart munitions involves more design requirements stemming from the addition of sensors and control mechanisms. The addition of these components must seek to minimize the weight and space impact on the overall projectile design so that desired target effects can still be achieved with the weapon. The inherent design conflict between standard projectile design considerations and new requirements imposed by sensors and control mechanisms has led designers to consider more complex geometric configurations. One such configuration is the differential roll projectile. This projectile configuration is comprised of forward and aft components. The forward and aft components are connected through a bearing, which allows the forward and aft portions of the projectile to spin at different rates. Figure 1 shows a schematic of the differential roll projectile configuration.

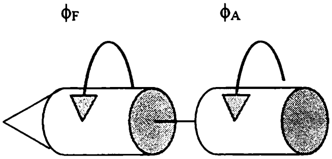


Figure 1 – Differential Roll Projectile Schematic

Typical flight mechanic analysis of a conventional, single-body, munition models the projectile with six degrees of freedom. Dynamic modeling of a differential roll projectile adds an additional roll degree of freedom to the equations of motion. This paper begins with the development of a dynamic model of a differential roll projectile in atmospheric flight including the additional roll degree of freedom. The model is derived such both the forward and aft bodies can be mass unbalanced. A hydrodynamic bearing couples the forward and aft components in the roll axis. The mathematical model is utilized to show trends in system response as a function of mass ratio and bearing resistance.

Differential Roll Projectile Dynamic Model

The mathematical model describing the motion of the differential roll projectile allows for 3 translation and 4 rotation rigid body degrees of freedom. The translation degrees of freedom are the three components of the mass center position vector. The rotation degrees of freedom are the Euler yaw and pitch angles as well as the forward body roll and aft body roll angles. The equations presented here use the ground surface as an inertial reference frame.[1]

Development of the kinematic and dynamic equations of motion is aided by the use of an intermediate reference frame. The sequence of rotations from the inertial frame to the forward and aft bodies consists of a set of body fixed rotations that are ordered: yaw, pitch, and forward/aft body roll. The fixed plane reference frame is defined as the intermediate frame before roll rotation. The fixed plane frame is common to both the forward and aft bodies.

Equations 1 are the translation kinematic differential equations that relate time derivatives of the mass center position components to the mass center velocity components in fixed plane reference frame.

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} c_\theta c_\psi & -s_\psi & s_\theta c_\psi \\ c_\theta s_\psi & c_\psi & s_\theta s_\psi \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{1}$$

Equations 2 are the rotation kinematic differential equations that relate time derivatives of the Euler angles with angular velocity components in the fixed plane reference frame.

$$\begin{Bmatrix} \dot{\phi}_F \\ \dot{\phi}_A \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_\theta \\ 0 & 1 & 0 & t_\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/c_\theta \end{bmatrix} \begin{Bmatrix} p_F \\ p_A \\ q \\ r \end{Bmatrix} \tag{2}$$

Equations 3 are the translation kinetic differential equations described in the fixed plane reference frame.

$$\begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} = \begin{Bmatrix} \frac{X}{m} \\ \frac{Y}{m} \\ \frac{Z}{m} \end{Bmatrix} - \begin{bmatrix} 0 & -r & q \\ r & 0 & 0 \\ -q & 0 & 0 \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{3}$$

Equations 4 are the rotation kinetic differential equations described in the fixed plane reference frame.

$$\begin{Bmatrix} \dot{p}_F \\ \dot{p}_A \\ \dot{q} \\ \dot{r} \end{Bmatrix} = [I]^{-1} \begin{Bmatrix} g_{F1} - M_V \\ g_{A1} + M_V \\ g_{F2} + g_{A2} \\ g_{F3} + g_{A3} \end{Bmatrix} \qquad (4)$$

A derivation of Equations 4 along with definitions of the right hand side components is provided in Appendices A and B.

As shown in Equations 5, the total applied force on the complete configuration is provided by the weight of both the forward and aft bodies (W) and air loads (A).

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{Bmatrix} X_W \\ Y_W \\ Z_W \end{Bmatrix} + \begin{Bmatrix} X_A \\ Y_A \\ Z_A \end{Bmatrix} \qquad (5)$$

The weight portion of the external loads is given by Equations 6,

$$\begin{Bmatrix} X_W \\ Y_W \\ Z_W \end{Bmatrix} = mg \begin{Bmatrix} -s_\theta \\ 0 \\ c_\theta \end{Bmatrix} \qquad (6)$$

while the aerodynamic force contribution is given by Equations 7.

$$\begin{Bmatrix} X_A \\ Y_A \\ Z_A \end{Bmatrix} = q_a D \begin{Bmatrix} C_{X0} + C_{XA2}\alpha^2 + C_{XB2}\beta^2 \\ C_{Y0} + C_{YB1}\beta \\ C_{Z0} + C_{ZA1}\alpha \end{Bmatrix} \qquad (7)$$

The longitudinal and lateral aerodynamic angles of attack are computed using Equations 8.

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right) \qquad \beta = \tan^{-1}\left(\frac{v}{u}\right) \qquad (8)$$

The aerodynamic coefficients in Equations 7 are functions of local Mach number at the projectile mass center. They are computed using linear interpolation from a table of data. The aerodynamic forces and moments are assumed to act solely on the forward body.

The right hand side of the rotation kinetic equations contains the externally applied moments on both the forward and aft bodies. The external moment components on the forward body are given by Equations 9 and contain contributions from steady (SA) and unsteady (UA) aerodynamics.

$$\begin{Bmatrix} L_A \\ M_A \\ N_A \end{Bmatrix} = \begin{Bmatrix} L_{SA} \\ M_{SA} \\ N_{SA} \end{Bmatrix} + \begin{Bmatrix} L_{UA} \\ M_{UA} \\ N_{UA} \end{Bmatrix} \qquad (9)$$

The steady body aerodynamic moment is computed by a cross product between the distance vector from the center of gravity to the center of pressure and the steady body aerodynamic force vector above. The unsteady body aerodynamic moment provides a damping source for projectile angular motion and is given by Equations 10.

$$\begin{Bmatrix} L_{UA} \\ M_{UA} \\ N_{UA} \end{Bmatrix} = q_a D^2 \begin{Bmatrix} C_{DD} + \dfrac{p_F D C_{LP}}{2V} \\ \dfrac{q D C_{MQ}}{2V} \\ \dfrac{r D C_{NR}}{2V} \end{Bmatrix} \qquad (10)$$

Air density is computed using the center of gravity position of the projectile in concert with the standard atmosphere.[2]

Simulation Example

In order to exercise the math model discussed above, consider a 6 feet long, a 120 pound projectile. The forward body is fin stabilized and the aft body is an internal circular cylinder. Aerodynamic forces and moments act on the forward body only. For this simulation set, initial forward body velocity is 750 m/s and initial gun elevation is 45 degrees. All other states variables are initially equal to 0. The projectile fins are canted slightly to provide a slowly rolling projectile in steady state.

Figures 2 through 10 show the state variables of the system versus time for the conditions mentioned above. The mass ratio of the aft body to the forward body is 1%. Under these circumstances, the projectile has a range of approximately 18 kilometers. Cross range, yaw angle, side velocity, vertical velocity, pitch rate, yaw rate, and aerodynamic angle of attack remain small throughout the event. Pitch attitude steadily decreases from 45 degrees to just below -60 degrees at impact. Figures 2 through 10 remain the same independent of the bearing resistance coefficient. Figures 12 and 13 show the roll angle and roll rate response as a function of bearing resistance coefficient. Values of bearing resistance coefficient are 0.000001, 0.000005, 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01 ft-lbf/rps. In Figure 12, the lowest trace is the response of the aft body for the lowest value of bearing resistance. The upper trace is the forward body roll response. For a bearing resistance coefficient of 0.00005, the aft body roll response is essentially the same as the forward body since both bodies rapidly couple in the roll axis. Figure 13 shows the roll rate trace for this simulation set. It is interesting that for lower values of bearing resistance, the aft body roll rate overshoots the forward body roll rate before settling.

Figures 14 and 15 show the roll angle and roll rate response of forward and aft bodies under the same conditions as the previous case except the mass ratio is now 50% rather than 1%. While the basic character of the roll response is the same, the aft body roll angle and hence roll rate build up relatively slowly due to the increase in aft body inertia.

Figures 16 through 19 show the response of the system with under the same conditions as Figures 14 and 15 except the initial roll rate of the aft body is – 100 rad/s. Like the other simulation results, lower values of bearing resistance produce slower roll response in the aft body. For larger splits in the forward and aft body roll rates, the trajectory begins to change as a function of bearing resistance owning to the fact that the roll response is sensitive to bearing resistance. In particular, Figure 17 shows the cross range under these circumstances. While the spray in the trajectory is only on the order of 10 meters, it points to the fact that if the forward and aft bodies possess substantially different initial roll rates, the trajectory becomes a function of bearing resistance.

Figures 19 through 22 show system response under the same conditions as Figures 16 through 19 except the mass ratio is now varied. Figure 22 shows the roll rate response. Due to aft body inertia changes, the roll response varies significantly with mass ratio. Subsequently, the trajectory begins to vary as well. Similar to the previous case, the trajectory spray is on the order of 10 meters and shows the trajectory of configurations with forward and aft bodies operating at significantly different roll rates is a function of the mass ratio.

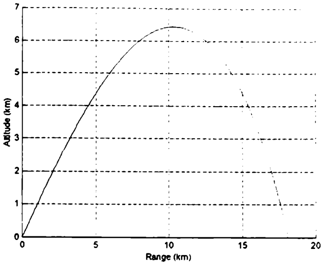


Figure 2 – Range

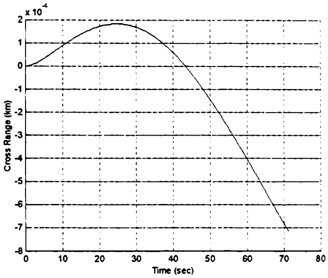


Figure 3 – Cross Range

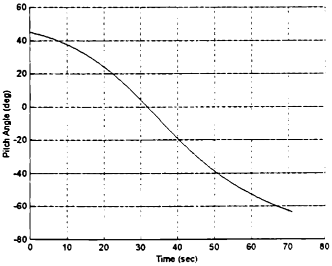


Figure 4 – Euler Pitch Angle

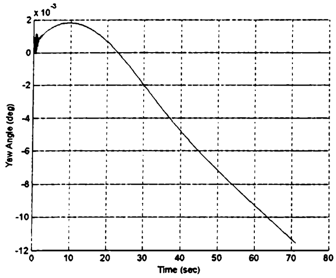


Figure 5 – Euler Yaw Angle

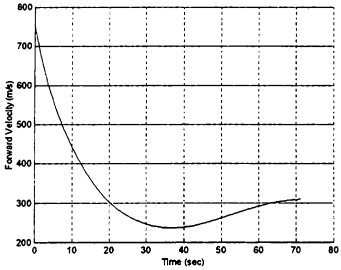


Figure 6 – Forward Body Velocity

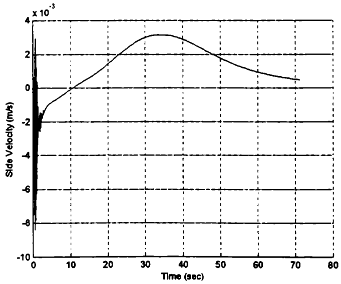


Figure 7 – Side Body Velocity

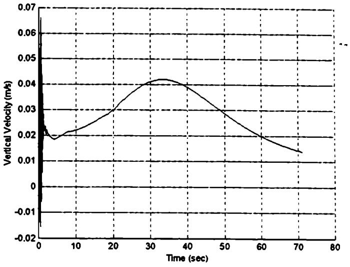


Figure 8 – Side Body Velocity

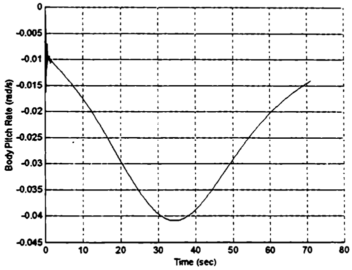


Figure 9 – Pitch Rate

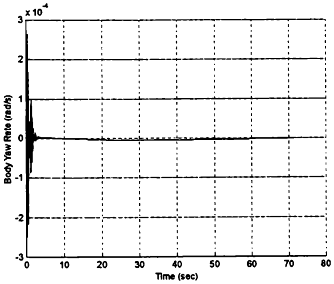


Figure 10 – Yaw Rate

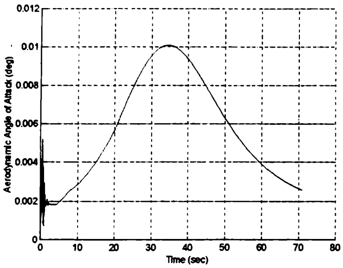


Figure 11 – Aerodynamic Angle of Attack

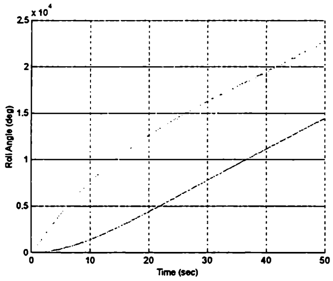


Figure 12 – Roll Angle (Mass Ratio = 1%, Damping Coefficient = 0.01-0.000001)

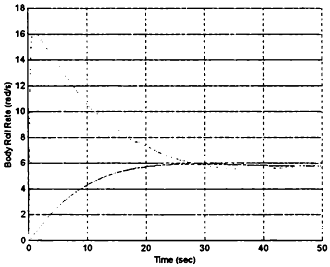


Figure 13 – Roll Rate (Mass Ratio = 1%, Damping Coefficient = 0.01-0.000001)

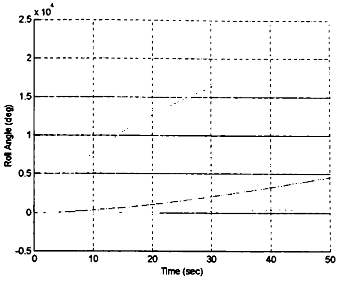


Figure 14 – Roll Angle (Mass Ratio = 50%, Damping Coefficient = 0.01-0.000001)

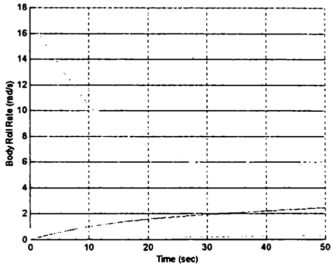


Figure 15 – Roll Angle (Mass Ratio = 50%, Damping Coefficient = 0.01-0.000001)

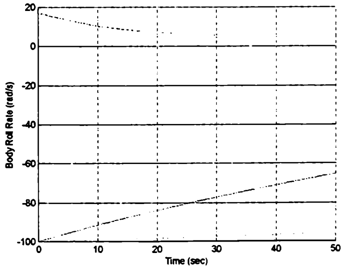


Figure 16 – Roll Rate (Mass Ratio = 50%, Damping Coefficient = 0.01-0.000001)

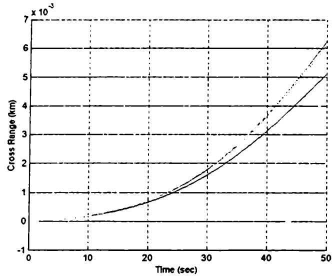


Figure 17 – Cross Range (Mass Ratio = 50%, Damping Coefficient = 0.01-0.000001)

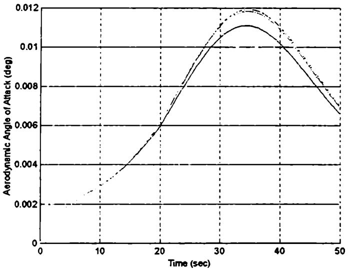


Figure 18 – Angle of Attack (Mass Ratio = 50%, Damping Coefficient = 0.01-0.000001)

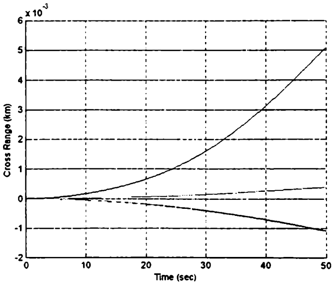


Figure 19 – Cross Range (Damping Coefficient = 0.0005, Mass Ratio = 1%-50%)

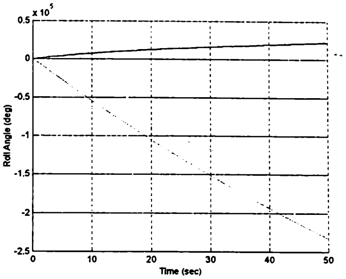


Figure 20 – Roll Angle (Damping Coefficient = 0.0005, Mass Ratio = 1%-50%)

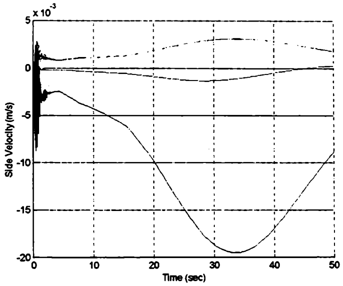


Figure 21 – Side Velocity (Damping Coefficient = 0.0005, Mass Ratio = 1%-50%)

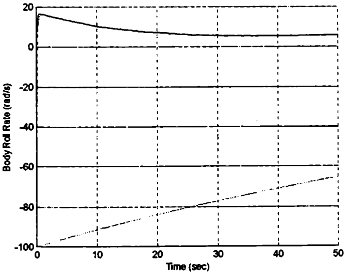


Figure 22 – Roll Rate (Damping Coefficient = 0.0005, Mass Ratio = 1%-50%)

Conclusions

The equations of motion for a differential roll projectile configuration with 7 degrees of freedom has been developed and exercised. The dynamic equations allow the forward and aft bodies to be mass unbalanced. A hydrodynamic bearing between the forward and aft components couples the roll degrees of freedom. Bearing resistance and forward/aft body mass ratio are the dominant factors in determining the roll dynamics. For spin rates typical of fin stabilized projectiles; the trajectory is essentially independent of both bearing resistance and mass ratio. However, for configurations with the forward and aft components operating at significantly different roll rates, the trajectory depends on the mass ratio and bearing resistance.

References

1.B. Etkin, Dynamics of Atmospheric Flight, John Wiley and Sons, New York, 1972.

2. R. Von Mises, Theory of Flight, Dover Publications Inc., New York, 1959.

3. C.M. Close, D.K. Frederick, Modeling and Analysis of Dynamic Systems, John Wiley and Sons, New York, 1995.

Appendix A – Constraint Forces and Moments

The rotation kinetic differential equations are derived by first splitting the two body system at the bearing connection point. Figures 2 and 3 show the external loads and internal constraint forces acting on both the forward and aft bodies, respectively.

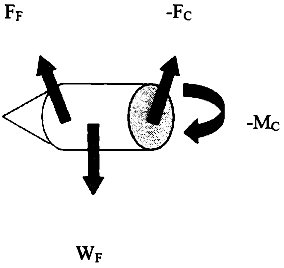


Figure 23 – Forces and Moments on the Forward Body

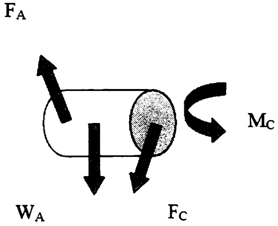


Figure 24 – Forces and Moments on the Aft Body

The constraint force, F_C, and the constraint moment, M_C, couple the forward and aft bodies. Key to the development of the rotation kinetic differential equations is the ability to solve for the constraint forces and moments as a function of state variables and time derivatives of state variables.

An expression for the constraint force can be obtained by subtracting the translation dynamic equations for both bodies.

$$\frac{m}{m_F m_A}\vec{F}_C = \frac{\vec{F}_F}{m_F} - \frac{\vec{F}_A}{m_A} + \vec{a}_{A/I} - \vec{a}_{F/I} \qquad (11)$$

The acceleration of the mass center of the forward and aft bodies, $\vec{a}_{F/I}$ and $\vec{a}_{A/I}$, can be expressed in terms of the acceleration of the composite body mass center. After making this substitution, the constraint force components in the fixed plane reference frame can be expressed in the following manner.

$$\begin{Bmatrix} F_{CX} \\ F_{CY} \\ F_{CZ} \end{Bmatrix} = \left[\overline{F}_F\right]\begin{Bmatrix} \dot{p}_F \\ \dot{q} \\ \dot{r} \end{Bmatrix} + \left[\overline{F}_A\right]\begin{Bmatrix} \dot{p}_A \\ \dot{q} \\ \dot{r} \end{Bmatrix} + \left\{\overline{F}_0\right\} \qquad (12)$$

The matrices $[\overline{F}_F]$, $[\overline{F}_A]$, $\{\overline{F}_0\}$ are complicated functions of the state variables and the geometry of the configuration.

The components in the fixed plane reference frame of the moment of the constraint force acting on the forward body about the forward body mass center can be written in the following manner.

$$\begin{Bmatrix} M_{FC_{FX}} \\ M_{FC_{FY}} \\ M_{FC_{FZ}} \end{Bmatrix} = \left[\overline{M}_{FF}\right]\begin{Bmatrix} \dot{p}_F \\ \dot{q} \\ \dot{r} \end{Bmatrix} + \left[\overline{M}_{FA}\right]\begin{Bmatrix} \dot{p}_A \\ \dot{q} \\ \dot{r} \end{Bmatrix} + \left\{\overline{M}_{F0}\right\} \qquad (13)$$

In a similar way, the components in the fixed plane reference frame of the moment of the constraint force acting on the aft body about the aft body mass center can be written in the following manner.

$$\begin{Bmatrix} M_{FC_{AX}} \\ M_{FC_{AY}} \\ M_{FC_{AZ}} \end{Bmatrix} = \left[\overline{M}_{AF}\right]\begin{Bmatrix} \dot{p}_F \\ \dot{q} \\ \dot{r} \end{Bmatrix} + \left[\overline{M}_{AA}\right]\begin{Bmatrix} \dot{p}_A \\ \dot{q} \\ \dot{r} \end{Bmatrix} + \left\{\overline{M}_{A0}\right\} \qquad (14)$$

The matrices $[\overline{M}_{FF}]$, $[\overline{M}_{FA}]$, $[\overline{M}_{AA}]$, $[\overline{M}_{AF}]$, $\{\overline{M}_{F0}\}$, and $\{\overline{M}_{A0}\}$ are also complicated functions of the state variables and the geometry of the configuration.

Appendix B – Rotation Kinetic Equations

The rotation kinetic differential equations are derived by first writing the Euler equations for the forward and aft bodies separately. These equations are both expressed in the fixed plane reference frame. The equations are left general and allow for a fully populated inertia matrix and mass unbalance. Equations 12, 13, and 14 are substituted into both sets of rotation kinetic equations for the forward and aft bodies. At this point both sets of equations still have unknown constraint moments at the bearing connection point. To eliminate the bearing constraint moments in the y and z direction in the fixed plane coordinate system, the y and z components of the rotation kinetic equations for the forward and aft bodies are added together to form 2 dynamic equation that are free of constraint moments. In this way, the constraint moments at the bearing have been eliminated analytically.

The forward and aft bodies are connected through a hydrodynamic bearing. The moment transmitted across a hydrodynamic bearing can be modeled as viscous damping.[3] The constitutive relation governing the constraint moment transmitted across a hydrodynamic bearing is given by Equation 15.

$$M_V = c_V\left(p_F - p_A\right) \qquad (15)$$

If the viscous damping coefficient, c_V, equals zero then the forward and aft body connection is frictionless.

The effective inertia matrix is a 4X4 matrix that is a combination of the inertia matrices of both the forward and aft bodies. As an aid in developing a formula for the effective inertia matrix, define the following intermediate matrices.

$$\left[I_{FF}\right] = \left[T_F\right]^T\left[I_F\right]\left[T_F\right] + \left[\overline{M}_{FF}\right] \qquad (16)$$

$$\left[I_{FA}\right] = \left[\overline{M}_{FA}\right] \qquad (17)$$

$$\left[I_{AA}\right] = \left[T_A\right]^T\left[I_A\right]\left[T_A\right] - \left[\overline{M}_{AA}\right] \qquad (18)$$

$$\left[I_{AF}\right] = -\left[\overline{M}_{AF}\right] \qquad (19)$$

Using Equations 16, 17, 18, and 19, elements of the effective inertia matrix can now be formed.

$$I_{1,1} = I_{FF_{1,1}} \qquad (20)$$

$$I_{1,2} = I_{FA_{1,1}} \qquad (21)$$

$$I_{1,3} = I_{FF_{1,2}} + I_{FA_{1,2}} \tag{22}$$

$$I_{1,4} = I_{FF_{1,3}} + I_{FA_{1,3}} \tag{23}$$

$$I_{2,1} = I_{AF_{1,1}} \tag{24}$$

$$I_{2,1} = I_{AA_{1,1}} \tag{25}$$

$$I_{2,3} = I_{AA_{1,2}} + I_{AF_{1,2}} \tag{26}$$

$$I_{2,4} = I_{AA_{1,3}} + I_{AF_{1,3}} \tag{27}$$

$$I_{3,1} = I_{FF_{2,1}} + I_{AF_{2,1}} \tag{28}$$

$$I_{3,2} = I_{AA_{2,1}} + I_{FA_{2,1}} \tag{29}$$

$$I_{3,3} = I_{FF_{2,2}} + I_{AA_{2,2}} + I_{FA_{2,2}} + I_{AF_{2,2}} \tag{30}$$

$$I_{3,4} = I_{FF_{2,3}} + I_{AA_{2,3}} + I_{FA_{2,3}} + I_{AF_{2,3}} \tag{31}$$

$$I_{4,1} = I_{FF_{3,1}} + I_{AF_{3,1}} \tag{32}$$

$$I_{4,2} = I_{AA_{3,1}} + I_{FA_{3,1}} \tag{33}$$

$$I_{4,3} = I_{FF_{3,2}} + I_{AA_{3,2}} + I_{FA_{3,2}} + I_{AF_{3,2}} \tag{34}$$

$$I_{4,4} = I_{FF_{3,3}} + I_{AA_{3,3}} + I_{FA_{3,3}} + I_{AF_{3,3}} \tag{35}$$

The elements of the right hand side vector are given by Equations 36 and 37.

$$\begin{Bmatrix} g_{F_1} \\ g_{F_2} \\ g_{F_3} \end{Bmatrix} = \begin{Bmatrix} M_{FX} \\ M_{FY} \\ M_{FZ} \end{Bmatrix} - \left[\overline{S}_F\right] \begin{Bmatrix} p_F \\ q \\ r \end{Bmatrix} - \left\{\overline{M}_{F0}\right\} \tag{36}$$

$$\begin{Bmatrix} g_{A_1} \\ g_{A_2} \\ g_{A_3} \end{Bmatrix} = \begin{Bmatrix} M_{AX} \\ M_{AY} \\ M_{AZ} \end{Bmatrix} - \left[\overline{S}_A\right] \begin{Bmatrix} p_A \\ q \\ r \end{Bmatrix} + \left\{\overline{M}_{A0}\right\} \tag{37}$$

The matrices $[\overline{S}_F]$ and $[\overline{S}_A]$ in Equations 36 and 37 are given by Equations 38 and 39.

$$\left[\overline{S}_F\right] = [T_F]^T [I_F][\dot{T}_F] + [T_F]^T [S_F][I_F][T_F] \tag{38}$$

$$\left[\overline{S}_A\right] = [T_A]^T [I_A][\dot{T}_A] + [T_A]^T [S_A][I_A][T_A] \tag{39}$$

where,

$$[S_F] = \begin{bmatrix} 0 & s_{\phi_F} q - c_{\phi_F} r & c_{\phi_F} q + s_{\phi_F} r \\ c_{\phi_F} r - s_{\phi_F} q & 0 & -p_F \\ -c_{\phi_F} q - s_{\phi_F} r & p_F & 0 \end{bmatrix}$$

$$[S_A] = \begin{bmatrix} 0 & s_{\phi_A} q - c_{\phi_A} r & c_{\phi_A} q + s_{\phi_A} r \\ c_{\phi_A} r - s_{\phi_A} q & 0 & -p_A \\ -c_{\phi_A} q - s_{\phi_A} r & p_A & 0 \end{bmatrix}$$

$$[T_F] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{\phi_F} & s_{\phi_F} \\ 0 & -s_{\phi_F} & c_{\phi_F} \end{bmatrix}$$

$$[T_A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{\phi_A} & s_{\phi_A} \\ 0 & -s_{\phi_A} & c_{\phi_A} \end{bmatrix}$$

$$[\dot{T}_F] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -(p_F + t_\theta r) s_{\phi_F} & (p_F + t_\theta r) c_{\phi_F} \\ 0 & -(p_F + t_\theta r) c_{\phi_F} & -(p_F + t_\theta r) s_{\phi_F} \end{bmatrix}$$

$$[\dot{T}_A] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -(p_A + t_\theta r) s_{\phi_A} & (p_A + t_\theta r) c_{\phi_A} \\ 0 & -(p_A + t_\theta r) c_{\phi_A} & -(p_A + t_\theta r) s_{\phi_A} \end{bmatrix}$$

AIAA-98-4544

CONSIDERATIONS ON SIMULATIONS TO VERIFY A SYSTEM CONCEPT FOR IMPROVED AIRPORT GUIDANCE

R.C. Meijer, M.Sc.EE,
Delft University of Technology, Faculty of Information Technology and Systems,
P.O. Box 5031, 2600 GA Delft, The Netherlands,
r.c.meijer@its.tudelft.nl

Abstract

Although properly equipped aircraft are capable of landing in conditions with nearly zero visibility, no onboard facilities are available yet, that guide the aircraft while vacating the runway and finding its way along the taxiways to the apron and the gate. As a result airport capacity reduces significantly in poor visibility conditions and queues may start to form in the air.

This paper introduces a research program which aims at demonstrating a concept to allow the pilot to move the aircraft more efficiently under low visibility conditions than by visual means only. This concept comprises the exocentric, spatially integrated presentation of the taxiway relative to the current aircraft position on the aircraft's Electronic Flight Instrument System.

The paper presents a preliminary system design including ground based and airborne systems. Furthermore, a simulation plan to verify assumptions, the concept and the associated system design is discussed. A low cost demonstrator is described which will be used to perform initial system verification. After this phase, implementation of the demonstrator in the Delft University laboratory aircraft is planned.

Introduction and Background

Nowadays, modern, properly equipped aircraft are capable of taking off in almost all weather conditions, fly through clouds, hail and rain without any visual cues followed by an approach and landing with zero visibility. However, the ground phase of the flight, taxiing, is affected most of all by reduced visibility conditions. This is a strange paradox, where flying, which is a relatively new mode of transport, can be done with the windows sealed, completely by instruments, whereas the oldest mode of transport, over ground, still needs visual cues to enable safe and expeditious transport from A to B.

The paradox is easily explained by the fact that air travel has to cope with frequent low visibility conditions within the demanding constraints of (fixed wing) aircraft whereas ground travel is not as frequently hindered by fog. Furthermore ground vehicles can easily reduce speed to cope with reduced visibility without crashing, as is not the case for (fixed wing) aircraft.

So, if flying aircraft can handle low visibility conditions, and taxiing aircraft cannot and we just stated that low visibility conditions on the ground are not as frequent as in the air, the question rises: how frequent do low visibility conditions on the ground occur? And above all, what is the definition of low visibility conditions in this context?

For the purpose of this paper only, low visibility conditions on the ground will be defined as the conditions where Category III landing operations are conducted. This means that the Runway Visual Range is 200 meter or less[2].

What happens when the visibility on and near the ground gets worse? For low visibility, a landing system with higher accuracy and integrity is required than for normal visibility. As the performance of the current landing system, the Instrument Landing System (ILS), is disturbed by leading aircraft, Air Traffic Control increases the separation between two arriving aircraft to guarantee the accuracy and integrity of the landing system, thus reducing the capacity of the runway. Furthermore, when aircraft have landed, the aircraft need to taxi from the runway to their allocated gate. Under low visibility conditions, Air Traffic Control does not have visual contact with traffic on the airport, which means that it must rely on (primary) radar images to perform its duty: control and guide aircraft. The fact that a primary radar requires that the Air Traffic COntroller

(ATCO) manually identifies every blip on the screen* implies a high mental workload for the ATCO. To reduce this workload, or to maintain a safe workload, the ATCO procedures require that the number of aircraft movements on the airport be reduced to only one or a few aircraft moving at the same time. This means that an ATCO handles less aircraft than under normal visibility conditions, or even less aircraft than the landing system allows. This results in another increase in separation between landing aircraft, further reducing the airport capacity. But what happens with aircraft en-route to the airport? It is clear that when the number of arriving aircraft is greater than the handling capacity of the airport, aircraft have to wait, flying holding patterns until ATC allows them to land. Currently, holding can be reduced by regulating take off based on the expected capacity on the arrival airport. Anyway, both holding in the air and on the ground causes significant delays whereas flying holding patterns also causes significant additional fuel consumption.

Fortunately, these conditions do not occur everywhere and on the places where they occur, they are not frequent. Furthermore, some airports do not handle aircraft during these conditions. Focusing on Europe for a moment, only a few airports handle aircraft during Cat III visibility conditions, notably: Amsterdam Airport, London Airports, Paris Airports and Frankfurt Airport. On Amsterdam airport Cat III conditions occur during approximately 9 days a year. For an airport with an annual number of movements of 380,000, this means that approximately 12,000 flights suffer from the low visibility conditions on Amsterdam alone. Although it is a tricky business translating this kind of figures into dollars, it will be clear that costs due to delays caused by low visibility conditions amount to millions of dollars yearly.

Not surprisingly, research in the United States and Europe aims at reducing these weather induced capacity problems. These research programs are mostly based on reducing delays by improving the efficiency of Air Traffic Control, especially Tower Control, without requiring additional equipment on board aircraft. The Taxi Navigation and Situational Awareness (T-NASA) and Roll-out Turn-off (ROTO) project, run by NASA[8], is an exception to this statement and uses a head-up display (HUD) to present guidance data. Another exception is the initiative from the Universities of Munich and Braunschweig to provide the pilot with high fidelity synthetic vision and a command display[5].

All three approaches have their advantages and disadvantages. The Air Traffic Control based programs do not provide track guidance based on which the pilot can actually steer the aircraft but aim at improved traffic management, i.e. preventing collisions and optimizing taxi routes. The NASA program requires HUDs in the aircraft, which are not very common for civil aircraft, implying a considerable investment for an aircraft operator. The German program requires a high performance graphics generator for a high fidelity synthetic vision system.

Another approach is being investigated in the Improved Airport Guidance (IMAGE) project at Delft University of Technology, the Netherlands. This project has recently been initiated and aims to demonstrate the feasibility of improving taxi performance by providing the pilot all necessary information for navigation, guidance and collision avoidance on the Electronic Flight Instrument System (EFIS). The guidance data is depicted as a perspective view using symbols as opposed to providing synthetic vision. As many airlines operate EFIS equipped aircraft and the taxi phase poses less stringent certification requirements on the EFIS, adding functionality to support surface navigation, guidance and control should be feasible with existing EFISs. To support the concept, an accurate positioning system, a datalink and some form of surveillance needs to be installed. If such an onboard system would be coupled to an Air Traffic Control system, pilots could be made aware of aircraft in the vicinity, stop bars and taxi clearances. The aim of this concept is to enable taxi speeds comparable to those under good visibility conditions using appropriately presented guidance data.

The paper will continue by identifying where solutions may be found to reduce either the cause or the impact of the capacity problem caused by taxiing in low visibility conditions. Potential solutions are briefly discussed to illustrate their merits and drawbacks. It will be shown that using the EFIS as a head down display to present navigation and guidance data promises to be a feasible and cost effective

* Primary Radar (PR) displays raw radar data. Secondary Surveillance Radar (SSR) interrogates a transponder in an aircraft and displays the identification of the aircraft next to its location on the radar display.

solution. The paper will then present a preliminary system design including ground based and airborne systems. Next, a simulation plan to verify assumptions, the concept and associated system design is discussed. Finally, a low cost demonstrator is described. This demonstrator will be used to perform initial system verification. After this phase, implementation in the Delft University laboratory aircraft is planned.

Potential Solutions

Starting from the problem formulation in the introduction, four different approaches were introduced to keep the airport capacity as high as possible during low visibility conditions. This section will investigate the fourth approach: providing the pilot with symbolic data for navigation, guidance and collision avoidance on EFIS displays. The questions to be answered are which data should be displayed and with which quality and quantity attributes. Following the answers to these questions, it should be determined which entity should provide and/or collect the data. As the project has just been started the data in the following sections can only be regarded as preliminary.

Pilot navigation point of view

Navigating on an airport can be quite a challenge, especially on large, often complex airports or on unfamiliar airports. On large, fast growing airports, the taxiway layout changes often and building works may prevent the use of certain tracks. Navigation is usually done by the pilot not flying using a sheet of paper showing a map of the airport. Low visibility conditions make it even more difficult for a pilot to determine the position of the aircraft on the airport since landmarks and signs are invisible or become apparent quite late. The result of these problems is that pilots are less confident in their position and tend to move slowly.

It is clear that the pilot may be helped here with two things:

First, an indication is needed of the position and orientation of the aircraft on the airport. This could be a map of the airport showing the aircraft position and orientation. This map should of course show runways, taxiways, aprons, gates and signs but also indications of forbidden areas.

Second, an indication is needed of where the aircraft is supposed to go. This may help the pilot to take the correct routes or, in other words, to prevent errors executing, often quite complex, taxi instructions. This indication could be projected as the route to be followed on the airport map.

Pilot collision avoidance point of view

It is apparent that low visibility conditions impair the visual detection of objects on the runway or taxiway by the pilot. This means that most important of all, other aircraft taxiing in the vicinity will be detected much later than under normal visibility conditions. But it also means that other airport vehicles or even luggage fallen off a trailer will be detected much later. This delayed detection of objects is normally compensated for by the pilot by reducing the aircraft speed, thus reducing the capacity of the taxiway.

In order to enable the pilot to use the same taxi speeds under limited visibility as under normal visibility, the pilot needs some way of detecting objects on the runway and taxiways, especially aircraft and airport vehicles. The fallen-off-luggage category should also be taken into account, although the probability of occurrence does not seem to be large.

Pilot guidance point of view

Anyone having driven a car in dense fog knows that keeping the car centered on the right lane is quite difficult. The same situation occurs when an aircraft is taxiing in low visibility conditions. Even though the pilot may know the aircraft's location and the location of every other vehicle or obstacle, a pilot will still have trouble keeping the aircraft on the centerline of the runway or taxiway. The risk of running off the concrete is normally compensated for by reducing speed.

To improve the pilot's tracking task, data needs to be presented on the desired track, cross track error, track angle error and the future track (prediction) to be able to track the centerline and steer through curves. One might consider also displaying exits and alike to give the pilot a preview of the situation to be expected.

ATCO point of view

Although not really the object of the IMAGE project, improving taxi performance in low visibility conditions cannot be considered completely separate from Air Traffic Control, more specifically, Tower Control.

Tower Control supervises all movements on the airport, from landing aircraft to airport vehicles, with the exception of aprons, which are managed by apron control. Supervising or controlling the vehicles can be modeled by an ever repeating three-step process starting with planning a route for every vehicle, issuing clearances to enable the routes and checking whether all aircraft stick to the clearances given. All these activities are supported by a surveillance function which provides position and identification data of all objects in the control area of the air traffic controller.

Under normal visibility conditions the air traffic controller can use his eyesight, sometimes enhanced by binoculars to perform the surveillance function. It is evident that under low visibility conditions eyesight needs to be enhanced, which is normally done by using a Primary Radar which scans the area. As a Primary Radar does not identify aircraft, contrary to a Secondary Surveillance Radar[†], the air traffic controller needs to keep a mental record of the identity of each blip on the screen as it moves. As this process is very attention demanding and error prone, ATC reduces the number of concurrent movements on the airport dramatically to maintain safety.

It is clear that even when aircraft should be able to taxi at normal speeds, ATC could still not allow the normal amount of concurrent movements keeping the taxi capacity low. Therefore, Air Traffic Control needs to be able to determine the location and identity of all aircraft, or more generally the objects in the control area, automatically. Note that this requirement is almost the same as the requirement formulated for the pilot's collision avoidance point of view.

From the Air Traffic Controller's guidance point of view, the ATCO needs to make a route plan for every aircraft, communicate it to that aircraft and make sure the route is without conflict. The ATCO's control task is then responsible for making sure the aircraft stick to the instructions they have received. The IMAGE project does not include research into route planning and conformance checking functions of ATC, but the errorless communication of the route and its clearances is of importance. Since the requirements for the pilot's navigation and guidance point of view indicate that some device presenting the desired route is needed, it must be ensured that ATC instructions are inserted into this device. One could propose to have the route instructions transferred by oral means whereas the pilot (not flying) inserts the instructions into the device. This is, however, error prone and time consuming. Preferably this process should be done automatically. This preference requires that the air traffic controller inserts the instructions, derived from a plan in his head, into a device, using a proper Man Machine Interface (MMI), which transmits the instructions to the aircraft's navigation device.

Preliminary System Design

Given the problems and their conceptual solutions introduced in the previous sections, numerous system solutions can be found which will or may solve the problems. However, several constraining conditions should be imposed on the potential solutions. Some obvious conditions are low cost, high capacity, ability to function with minimal change in infrastructure, ability to function when not all aircraft participate, unambiguous, etc. Taking these conditions into account still leaves quite some combinations. This section identifies conceptual solutions for the problems identified before, focused on the aircraft and on communication between ATC and aircraft.

Aircraft Positioning and Navigation

First of all, since the system to be designed is to help the pilot, the aircraft functions will be considered. It was concluded that the pilot needs to know the position, orientation and speed of the aircraft. These parameters can be determined in many ways with many different systems. In fact, it is not very important which systems to use for this function, but what value its Required Navigation Performance (RNP) parameters should have[3,4]. The RNP concept encompasses accuracy, integrity, continuity and availability. For guiding an aircraft, the bandwidth of positioning, orientation and speed indication is important as well. In order to help the pilot guide the aircraft, it may be necessary to provide a prediction of the aircraft state some time ahead.

[†] Ordinary Secondary Surveillance Radar cannot be used on airports because several aircraft may answer an interrogation at (almost) the same time, resulting in garbled answers.

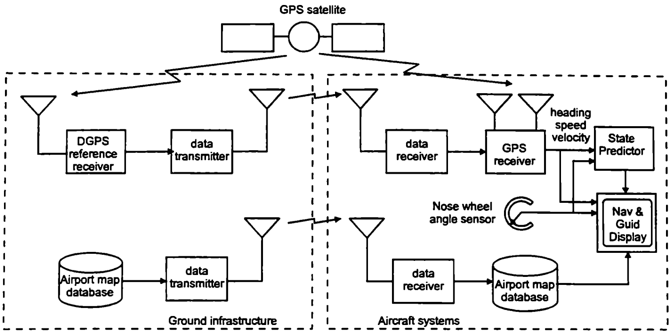


Figure 1 Positioning and Navigation for taxiing

Although these requirements have not been quantified yet, a first estimates indicates that the requirements are far less stringent than for landing aircraft, since navigation and guidance during the taxi phase is considered less critical. Reasons are lower speed, allowing greater detection time, and lower damage in the event of an accident.

Taking these points into account one could envisage an onboard DGPS system to be used for positioning, heading and speed determination [‡] plus a local DGPS reference station. For the distribution of DGPS a local radio broadcast system is needed. This may be a DGPS station already in use for other phases of flight, depending on the performance characteristics offered[1]. For the prediction of the aircraft position, a sensor measuring the angle of the nose wheel may be used.

Now that the navigation system is chosen, a map showing runways, taxiways, aprons and gates, is needed to project the aircraft position on enabling the pilot to navigate on the airport. The map needs to be stored electronically, but not necessarily permanently. One may choose to store the map for example on a disk, which is update every few weeks, just like navigation data for Flight Management Systems are updated. However, this has an inherit weak integrity. Collecting large quantities of data, processing and distributing it, are still not 100% reliable. Current figures indicate an error in databases for 1 in 10,000 items. This may of course be sufficient for the taxi application. One may also consider to up-link airport data from Air Traffic Control to the aircraft. This option circumvents certain stages of the process inherent to the disk option and may be more reliable. Other advantages are that the database is always up to date and that there is no need to carry around a database with airport maps.

The positioning and navigation functions are illustrated by Figure 1.

Surveillance data collection and distribution

The second requirement indicated that a pilot needs to be able to detect vehicles and other objects in the vicinity. Air Traffic Control also needs to know where every vehicle is located but it needs to know the identity of every vehicle as well. The pilot and ATC requirements have such a large overlap that it will probably be worthwhile to find a solution, which fits both surveillance requirements.

The pilot requirements could be satisfied by having each aircraft scan its surroundings and display the situation to the pilot. Another option, satisfying the pilot requirements, is to have Air Traffic Control scan the airport and transmit (a compressed or interpreted)

[‡] It still has to be determined whether a two antenna DGPS system could provide heading and speed with sufficient performance characteristic. These items could also be determined by a magnetic (compass) system and speedometers measuring the revolutions of the wheels.

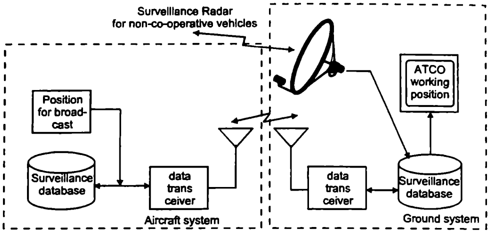


Figure 2 Surveillance functions for taxiing

picture of the situation. This option could be modified to satisfy the ATC requirements as well by using a Secondary Surveillance Radar (SSR) to establish the identity of vehicles. Yet another option is to have all vehicles transmit their position and identity to everybody, including ATC.

The first option requires complicated and therefore expensive sensors in every aircraft like millimeter wave radar or infrared equipment. It is self sufficient, but does not satisfy the ATC requirements.

The second option requires ATC to have a high performance radar system. As a radio data broadcast system is needed to transmit the radar image to the aircraft, compression or interpretation of the radar image is necessary to ensure efficient use of radio frequency (RF) bandwidth as it is a scarce resource. Using a Secondary Surveillance Radar will make the interpretation of the radar image easier if garbling of replies could be prevented (for example by selective interrogation). Note that this option requires infrastructure on the airport.

The third option requires all vehicles to co-operate and transmit their position regularly. An aircraft then receives position updates of every other aircraft or vehicle located nearby. This means, however, that this option will be unable to locate fallen-off luggage. Having all aircraft transmit their position may be unreliable and may be an inefficient use of RF bandwidth unless regulated in some way.

The third option, combined with the second is however quite attractive. Every co-operating aircraft transmits its position regularly, which is detected by ATC, which transmits the positions of non co-operating aircraft, vehicles and other obstacles. The aircraft transmissions could either be done in an unregulated transmit-as-you-please way, resulting in some messages getting lost due to aircraft transmitting at the same time, or in a regulated way, requiring some regulation algorithm. One may think of using the Automatic Dependent Surveillance Broadcast (ADS-B) which has recently been developed. An advantage of this option is the ability of the system to degrade gracefully also enabling a transition path. A disadvantage of this implementation of the surveillance function is that both an aircraft and an air traffic control system are necessary.

The surveillance data acquisition and distribution is illustrated in Figure 2.

Command distribution

The third function to be implemented is the distribution of commands from Air Traffic Controller to the pilot and the aircraft systems, where commands include a route, clearances, etc. Commands are specifically intended for one specific aircraft and need therefore be addressed. Furthermore, commands need to be acknowledged by the pilot[§], contrary to surveillance, positioning data and map data. And finally, as mentioned before, the commands should be in a machine-readable format, to minimize the probability of misinterpretation. This includes using waypoints and landmarks by reference from the

[§] Commands are distributed using a data-link (contrary to data-broadcast) which means that when errors are detected, the receiver requests a re-transmission. When data is received correctly, the receiver will send an acknowledgment. This acknowledgment is on a lower OSI level than the pilot acknowledgment.

airport map, which is assumed present in the aircraft (either by disk or by upload).

In order to keep the system simple, the command set is kept small for the taxi system. The commands include a proposed taxi route, which may be accepted or rejected by the pilot, and clearances, which must be acknowledged by the pilot[¶]. All other dialogues would still have to be done by radiotelephony.

Aircraft Display and Command systems

Now that the functions and allocation of functions are clear, the aircraft display and command systems will be considered (ATCO display and command systems will not be considered in this paper).

The pilot needs to navigate, guide the aircraft and avoid collisions. These functions have more or less overlapping requirements. Theunissen found that: "An exocentric frame of reference can be used to combine a high position error gain with adequate trajectory preview ... " [7], where exocentric is to indicate a frame of reference with a viewpoint located behind and above the pilot position. For the navigation task, however, a plan view display will be most convenient as a replacement for the paper airport map.

So two displays (or at least two views) are necessary to accommodate the functions the pilot must perform. The IMAGE project proposes to use the Primary Flight Display (PFD) and the Navigation Display (ND) for the exocentric view and plan view respectively, since these display systems are available in almost any modern aircraft and are not used anymore as soon as the aircraft has landed. This additional display functionality can be implemented by adding software to the Display Electronics Unit (DEU). As soon as an aircraft has landed, the mode of the DEU needs to be changed (by the Display Control Unit) to display the perspective and plan view instead of PFD and ND. This concept is illustrated in Figure 3. The re-use of equipment proposed here will help to minimize the cost of the proposed concept.

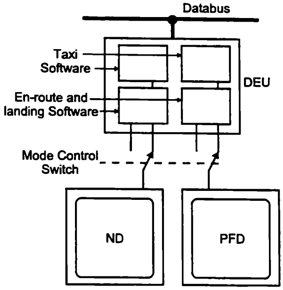


Figure 3 EFIS enhanced with taxi display software

As soon as the DEU is set to the taxi-mode, commands can be received and displayed on the Command Display Unit (CDU), the exocentric and the plan view display. Acknowledgments to commands can be send by the pilot using the CDU or by another pilot input device.

Details on the design of an exocentric frame of reference are presented by Theunissen[7].

Simulation plan

The ultimate goal of the IMAGE project is, as indicated before, to demonstrate the feasibility of providing improved information for the pilots to allow them to taxi more efficiently, especially under low visibility conditions. It is intended that this goal is reached by a process of gradually implementing, testing and simulating the functionality; a spiral based approach. This approach is chosen to ensure that the concept can be refined during the stages of the development, gradually obtaining increasing confidence in the feasibility of the concept.

This section will describe six modeling and simulation stages, which should lead to the final demonstrator, built into an aircraft.

[¶] It is recognized that other datalink commands are being investigated that promise to give large improvements in efficiency, e.g. startup clearances. However, since the purpose of this project is to demonstrate the improved efficiency of using perspective displays for taxiing under low visibility conditions, only the datalink commands pertinent to that goal will be implemented.

Part task experiments

In the first stage the guidance, collision avoidance and navigation concept is to be implemented and tested using a part task experiment. These experiments involve a pilot working position with a perspective display, presenting guidance and collision avoidance data. No interaction with Air Traffic Control is applied during this phase of the simulation.

The test person is to maintain the aircraft on the runway and taxiways following the route presented to him. This experiment should provide data to determine the best position of the exocentric viewpoint in terms of guidance, like preview, lag, resolution and accuracy. The collision avoidance task should provide data to determine the preview, field of view and update rate of surveillance data to be able to detect obstacles and signs.

The initial results of the first stage experiments are reported by Theunissen[7].

During the second stage, the plan view display design should be tested.

In the third stage the interaction with Air Traffic Control is to be simulated in a part task experiment. This involves adding a data-link simulation, dialogues and extra in- and output to the pilot. The experiment should provide data on the amount of work introduced by data-link interaction and the appropriate implementation of these interactions. To enable this stage, an Air Traffic Control working position needs to be created which will just fit the needs for the experiments. The surveillance data needed for the Air Traffic Controller will be provided by the aircraft, as it would be done in real-life by an ADS-B system. Other traffic and obstacles are generated by the Air Traffic Controller working position, which also transmits their position to the aircraft, as it would do in real-life.

During stage four, the simulated data-link will be replaced by a real data-link resembling the VHF Data Link (VDL), which is one of the data-links that has been proposed for aeronautical data-link applications. The reason for choosing a link *resembling* VDL is one of economy, since VHF data-radios are available, notably in the amateur radio market, for reasonable prices and good support. This choice will also ensure that the experiments will not suffer from certain effects in the experiment that would never happen in real life. The bandwidth, for example, of the amateur VHF data-radios for a radio wavelength of 2 meters normally limits the data-rate to 9600 bit per second. Furthermore, the radio wave propagation effects like attenuation, shadowing and multi-path will be similar in a 2 meter experimental data-link to that of the VDL.

During this stage similar experiments will be performed as in the third stage. If possible, simulating several other aircraft using the radio data-link to introduce competition on the radio data-link.

Low cost vehicle

At this point of the experiments, quite some confidence should have been built up to support the concept and the implementation so far. In the next stage, stage five, the pilot working position will be built into a low cost road vehicle. The goal of this stage is not to do a Man Machine Interface experiment, since the vehicle dynamics and handling will be quite different from an aircraft, but to test all systems. Experiments using this setup will also provide data on the usability of the radio data-link under various conditions (line of sight, around corners, in shadowing conditions, etc), which values depend on the modulation and correction techniques used in the experimental data-link as compared to the VDL. The moving vehicle will also require the implementation of a positioning system, most probably a Differential GPS device using a DGPS reference station located nearby. A dual DGPS system or a magnetic compass may be used to determine the heading of the aircraft. To enable the vehicles position prediction, a sensor may need to be mounted to indicate the angle of the wheels relative to the vehicle body.

The experiments using the vehicle will commence with a driver and co-driver, each being able to look outside the vehicle and to the perspective display. When enough confidence has been acquired regarding steering the vehicle, Air Traffic Control interaction will be added to the experiment introducing a desired route, other vehicle, obstacles and road-signs.

The results of these experiments should accumulate enough confidence to go to stage six in which the actual demonstrator will be build and experimented with.

Demonstrator

The actual demonstrator for this project will be built into the University's Cessna Citation II. Most devices and instruments used for the low cost vehicle experiments will be remounted into aircraft without change. The pilot will be provided with a display to overlap the existing display (which may not be tampered with to maintain certification) to be shown the perspective view. The wheel angle sensor will now be mounted on the nose wheel and the DGPS systems will use antennas on the aircraft.

The experiments will first collect data on the track keeping of the pilot using the perspective display, while the pilot not flying will keep a look out of the window. As soon as confidence has been gained, in this set-up, air traffic control interaction is added, introducing a commanded route, clearances, signs, objects and other vehicles.

Being confident that the experiments are successful, the set-up will be used to demonstrate the concept to interested and invited parties from within the country and abroad.

In Conclusion

The IMAGE concept is aimed at enabling increased taxi speeds under low visibility conditions. Even though other projects aim at the same goal, the IMAGE concept is unique in its proposed implementation. The concept proposes to use the displays of the Electronic Flight Instrument System to show an exocentric presentation of the taxiways relative to the current aircraft position and a plan view display for navigation. A demonstrator will be build to show the feasibility of the concept using only limited resources. Some of the work has already started; in particular preliminary part task experiments have been executed and proved successful.

Funding for this project has been requested at the Dutch Technology Foundation STW.

References

1. **GNSSP,** *Draft SARPS for GBAS,* Global Navigation Satellite Systems Panel, working group D meeting, draft version 0.7, Montreal, June 1997.

2. **ICAO,** *Manual of all-weather operations,* International Civil Aviation Organization, Doc 9365-AN/910, second edition, Montreal, 1991

3. **ICAO,** *Manual on Required Navigation Performance,* International Civil Aviation Organization, Doc 9613-AN/937, first edition, Montreal, 1994.

4. **Kelley, R.J., Davis, J.M.,** *Required Navigation Performance (RNP) for Precision Approach and Landing with GNSS Application,* Journal of The Institute of Navigation, Vol. 41, No. 1, Spring 1994.

5. **Möhlenkamp, K., Schänzer, G.,** *Automatic Control Steps for Aircraft Taxi Guidance,* AGARD CP 538, pp 28-1 - 28-5, 1993.

6. **Sachs, G., Möller, H. Dobler, K., Schänzer G., Möhlenkamp, K.,** *Synthetic Vision and Precision Navigation for aircraft taxi guidance in low visibility,* Proceedings of the AIAA Guidance, Navigation and Control Conference, pp. 1202-1211, Scottsdale, AZ., 1994.

7. **Theunissen, E.,** *Structured specification of exocentric frames of reference,* AIAA Modeling and Simulation Technology Conference, Boston, 1998.

8. **Young, Steven D. and Jones, Denise R.,** *Flight Testing of an Airport Surface Guidance, Navigation and Control System,* ION National Technical Meeting, January 21-23, 1998.

AUTHOR INDEX

AIAA

For permission to copy or republish, contact:
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive
Suite 500
Reston, VA 20191-4344

ISBN
1-56347-271-6